

Действителен: с 28.05.2026 по 28.08.2027

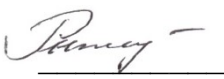
8.207 «Российский новый университет»

Елец
2026 год

Одобрена
предметной (цикловой)
комиссией общеобразовательных
дисциплин

Разработана на основе Федерального
Государственного Образовательного
Стандарта Среднего
Профессионального Образования/
ФГОС СПО/ по специальности 09.02.12
Техническая эксплуатация и
сопровождение информационных
систем, утвержденного приказом
Министерства просвещения РФ от
10.03.2025 г. N 184

Протокол № 5
от «28» 01 2026 г.
Председатель предметной
(цикловой) комиссии
 / С.В. Толоконников

Начальник отделения СПО
 / О.В. Рыжкова

Составитель (автор):  / пр. С.В. Толоконников

Рецензент: Суздальская Ева Анатольевна, заместитель начальника отдела
эксплуатации средств вычислительной техники, программного обеспечения и
телекоммуникационного оборудования управления информационных
технологий – Отделение Социального фонда Российской Федерации по
Липецкой области, г. Елец

1. НАИМЕНОВАНИЕ И ЦЕЛЬ ОСВОЕНИЯ ДИСЦИПЛИНЫ

Программа учебной дисциплины Основы алгоритмизации и программирования предназначена для изучения основ программирования в профессиональных образовательных организациях, реализующих образовательную программу среднего общего образования в пределах освоения образовательной программы СПО на базе основного общего образования.

Программа разработана на основе Федерального Государственного Образовательного Стандарта Среднего Профессионального Образования/ ФГОС СПО/ по специальности 09.02.12 ТЕХНИЧЕСКАЯ ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ИНФОРМАЦИОННЫХ СИСТЕМ (приказ Минобрнауки России от 10.03.2025 г. № 184).

Учебная дисциплина Основы алгоритмизации и программирования изучается в общепрофессиональном цикле учебного плана ОП на базе основного общего образования с получением среднего общего образования. В учебных планах учебная дисциплина Основы алгоритмизации и программирования входит в состав профессиональных учебных дисциплин ФГОС среднего общего образования.

Целью учебной дисциплины является формирование у обучающихся фундаментальных знаний, практических навыков и профессиональных компетенций в области алгоритмизации и программирования, необходимых для разработки, анализа и реализации программных решений.

Изучение дисциплины направлено на подготовку обучающихся к осуществлению профессиональной деятельности, выполнению обобщённых трудовых функций, трудовых функций, трудовых действий, определённых профессиональным стандартом, соответствующим ОП.

Развитие у обучающихся навыков командной работы, межличностной коммуникации, принятия решений, лидерских качеств обеспечивается чтением лекций, проведением занятий с применением активных методов обучения, содержание которых разработано на основе результатов научных исследований, проводимых Университетом, в том числе с учетом региональных особенностей профессиональной деятельности выпускников и потребностей работодателей.

2. ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫЕ С ПЛАНИРУЕМЫМИ РЕЗУЛЬТАТАМИ ОСВОЕНИЯ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ

В результате освоения дисциплины обучающийся по программе СПО по специальности должен:

- Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам; (ОК 01.)

Знать:

- понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции.

Уметь:

- разрабатывать алгоритмы для конкретных задач.

- Использовать современные средства поиска, анализа и интерпретации информации и информационные технологии для выполнения задач профессиональной деятельности; (ОК 02.)

Знать:

- эволюцию языков программирования, их классификацию, понятие системы программирования.

Уметь:

- использовать программы для графического отображения алгоритмов. Определять сложность работы алгоритмов.

3. ОБЪЕМ ДИСЦИПЛИНЫ В ЗАЧЕТНЫХ ЕДИНИЦАХ С УКАЗАНИЕМ КОЛИЧЕСТВА АКАДЕМИЧЕСКИХ ЧАСОВ, ВЫДЕЛЕННЫХ НА КОНТАКТНУЮ РАБОТУ ОБУЧАЮЩИХСЯ С ПРЕПОДАВАТЕЛЕМ (ПО ВИДАМ УЧЕБНЫХ ЗАНЯТИЙ) И НА САМОСТОЯТЕЛЬНУЮ РАБОТУ ОБУЧАЮЩИХСЯ

Общая трудоемкость дисциплины составляет 5 зачетных единиц (180 час.).

Вид учебной работы	Объем часов
Максимальная учебная нагрузка (всего):	180
Обязательная аудиторная учебная нагрузка (всего)	156
в том числе:	
Лекционные занятия (Лек)	48
Практические занятия (Пр)	102
Контрольная работа (КоРаб)	0,3
Консультации (Конс)	2

Экзамен (Э)	0,4
Текущие консультации (ТКонс)	3,3
Самостоятельная работа обучающегося (всего)	8,4
Внеаудиторная самостоятельная работа	15,6

**Распределение учебного времени по темам и видам учебных занятий
очная форма обучения**

№	Наименование разделов, тем учебных занятий	Всего часов			Контактная работа с преподавателем					Сам. раб.	Формируемые результаты обучения
			Всего	Л	Пр	Э	К	КоР	Тконс		
Основы алгоритмизации и программирования											
1.	Основы алгоритмизации, языки и системы программирования.	10	10	4	6						ОК 01., ОК 02.
2.	Первое знакомство с языком программирования Python.	12	10	4	6					2	
3.	Строки и операции над строками.	12	12	2	10						
4	Модули и пакеты.	10	10	2	8						
5	Строковые методы. Списки.	12	10	2	8					2	
6	Итерации.	16	14	4	10					2	
7	Промежуточная аттестация. Контрольная работа.	2	2					0,3	1,7		

8	Множества, кортежи, словари.	16	16	6	10					
9	Реализация алгоритмов.	10	10	4	6					
10	Обработка исключений в Python.	10	10	4	6					
11	Работа с файлами.	12	12	4	8					
12	Объектно-ориентированное программирование на языке Python.	12	12	4	8					2,4
13	Итераторы, генераторы и декораторы.	8	8	4	4					
14	Разработка приложений с графическим	18	18	6	12					
15	Промежуточная аттестация. Экзамен.	4	4			0,4	2		1,6	

4. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ, СТРУКТУРИРОВАННОЕ ПО ТЕМАМ (РАЗДЕЛАМ)

Тема 1. Основы алгоритмизации, языки и системы программирования

Основы алгоритмизации. Алгоритмы цикла. Основы алгоритмизации. Языки и системы программирования. Алгоритмы. Языки программирования

Тема 2. Первое знакомство с языком программирования Python

Интеллектуальный калькулятор. Переменные в языке Python. Функции в языке Python. Программы в отдельном файле. Область видимости переменных. Применение функций. Схема передачи чисел в качестве аргументов функции.

Тема 3. Строки и операции над строками.

Операции над строками. Дополнительные возможности функции print. Ввод значений с клавиатуры. Логические выражения. Условная инструкция if. Строки документации

Тема 4. Модули и пакеты

Встроенные модули и пакеты. Создание собственных модулей. Автоматизированное тестирование функций. Генерация псевдослучайных чисел

Тема 5. Строковые методы. Списки.

Строковые методы. Операции над списками. Псевдонимы и копирование списков. Методы списка. Преобразование типов. Схема передачи списков в качестве аргументов функции. Построение графиков из списков.

Тема 6. Итерации.

Инструкция for. Функция range. Создание списка. Вложенные списки. Инструкция while. Массивы NumPy и работа с Матрицами

Тема 7. Промежуточная аттестация. Контрольная работа.

Решение конкретной теоретической или практической задачи для установления степени усвоения изучаемого материала студентом.

См. приложение 1 «Контрольная работа»

Тема 8. Множества, кортежи, словари.

Множества. Кортежи. Словари.

Тема 9. Реализация алгоритмов.

Реализация алгоритмов

Тема 10. Обработка исключений в Python.

Обработка исключений в Python

Тема 11. Работа с файлами.

Байтовый тип данных. Регулярные выражения. Работа с форматом JSON.
Пакеты для анализа данных на языке Python.

Тема 12. Объектно-ориентированное программирование на языке Python.

Основы объектно-ориентированного подхода. Наследование классов.

Тема 13. Итераторы, генераторы и декораторы.

Итераторы и генераторы. Декораторы.

Тема 14. Разработка приложений с графическим интерфейсом.

Основы работы с модулем tkinter. Шаблон «модель — вид — контроллер» на примере модуля tkinter. Изменение параметров по умолчанию при работе с tkinter.

Тема 15. Промежуточная аттестация. Экзамен.

Решение конкретной теоретической или практической задачи для установления степени усвоения изучаемого материала студентом.

См. приложение 2 «Вопросы к экзамену»

Планы практических занятий

Тема 1. Основы алгоритмизации, языки и системы программирования.

Время - 6 час.

Основные вопросы: Основы алгоритмизации. Алгоритмы цикла. Основы алгоритмизации. Языки и системы программирования. Алгоритмы. Языки программирования.

См. Приложение 3 «Практические работы».

Тема 2. Первое знакомство с языком программирования Python..

Время - 6 час.

Основные вопросы: Интеллектуальный калькулятор. Переменные в языке Python. Функции в языке Python. Программы в отдельном файле. Область видимости переменных. Применение функций. Схема передачи чисел в качестве аргументов функции. Создание консольного приложения «Калькулятор».

См. Приложение 3 «Практические работы».

Тема 3. Строки и операции над строками.

Время - 10 час.

Основные вопросы: Операции над строками. Дополнительные возможности

функции print. Ввод значений с клавиатуры. Логические выражения. Условная инструкция if. Строки документации. Создание консольного приложения с вводом значений с клавиатуры. Создание программ с выводом текста. Создание программ с применением условной инструкции if..

См. Приложение 3 «Практические работы».

Тема 4. Модули и пакеты.

Время - 8 час.

Основные вопросы: Встроенные модули и пакеты. Создание собственных модулей. Автоматизированное тестирование функций. Генерация псевдослучайных чисел. Подключение модулей. Создание собственного модуля. Использование модулей для консольного приложения

См. Приложение 3 «Практические работы».

Тема 5. Строковые методы. Списки.

Время - 8 час.

Основные вопросы: Использование строковых методов для консольного приложения. Создание программ на преобразование элементов списка

См. Приложение 3 «Практические работы».

Тема 6. Итерации.

Время - 10 час.

Основные вопросы: Инструкция for. Функция range. Создание списка. Вложенные списки. Инструкция while. Массивы NumPy и работа с Матрицами. Создание программ с циклической структуры Создание программ с применением функции range. Составление программ для создания списка. Составление программ для создания вложенных списков. Составление программ для создания матриц.

См. Приложение 3 «Практические работы».

Тема 8. Множества, кортежи, словари.

Время - 10 час.

Основные вопросы: Множества, кортежи, словари. Составление программ с применением множеств для решения задач.

См. Приложение 3 «Практические работы».

Тема 9. Реализация алгоритмов.

Время - 6 час.

Основные вопросы: Реализация алгоритмов. Составление программ с реализацией стека.

См. Приложение 3 «Практические работы».

Тема 10. Обработка исключений в Python.

Время - 6 час.

Основные вопросы: Составление программ с реализацией исключений. .

См. Приложение 3 «Практические работы».

Тема 11. Работа с файлами.

Время - 8 час.

Основные вопросы: Байтовый тип данных. Регулярные выражения. Работа с форматом JSON. Пакеты для анализа данных на языке Python.. Составление программ на создание текстового файла для решения задач. Создание программ для работы с каталогами и файлами.

См. Приложение 3 «Практические работы».

Тема 12. Объектно-ориентированное программирование на языке Python.

Время - 8 час.

Основные вопросы: Основы объектно-ориентированного подхода. Наследование классов. Программирование простейших классов. Классы ООП: виды, назначение, свойства, методы, события.

См. Приложение 3 «Практические работы».

Тема 13. Итераторы, генераторы и декораторы.

Время - 4 час.

Основные вопросы: Итераторы и генераторы. Декораторы. Составление программ с реализацией итератора

См. Приложение 3 «Практические работы».

Тема 14. Разработка приложений с графическим интерфейсом.

Время - 12 час.

Основные вопросы: Основы работы с модулем tkinter. Шаблон «модель - вид — контроллер» на примере модуля tkinter. Изменение параметров по умолчанию при работе с tkinter. Введение в разработку графических пользовательских интерфейсов. Разработка интерфейса приложения. Разработка подсистемы регистрации и авторизации для автоматизированной информационной системы. Разработка минимального функционала автоматизированной информационной системы.

См. Приложение 3 «Практические работы».

5. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ

5.1.1. Методические указания:

Самостоятельная работа обучающихся, это работа, выполняемая во внеаудиторное (аудиторное) время при методическом руководстве преподавателя с целью овладения необходимыми компетенциями по дисциплине, дополнительными знаниями, умениями и навыками, опытом творческой, исследовательской деятельности, развития самостоятельности, ответственности и организованности, творческого подхода к решению проблем учебного уровня.

Видами заданий для самостоятельной работы являются:

1. Заполнение глоссария
2. Поиск и анализ источников и литературы
3. Подготовка сообщений, презентаций, докладов, рефератов
4. Выполнение практических работ
5. Решение задач

Для выполнения заданий по самостоятельной работе рекомендуется использовать материалы учебников и интернет ресурсов:

Основная литература:

1. Кудрина, Е. В. Основы алгоритмизации и программирования на языке С#: учебник для среднего профессионального образования / Е. В. Кудрина, М. В. Огнева. — Москва : Издательство Юрайт, 2026. — 322 с. — (Профессиональное образование). — ISBN 978-5-534-10772-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/587050>

2. Паронджанов, В. Д. Алгоритмические языки и программирование: ДРАКОН: учебник для вузов / В. Д. Паронджанов. — Москва : Издательство Юрайт, 2026. — 436 с. — (Высшее образование). — ISBN 978-5-534-13146-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/588170>

3. Трофимов, В. В. Основы алгоритмизации и программирования: учебник для среднего профессионального образования / В. В. Трофимов, Т. А. Павловская. — 4-е изд. — Москва : Издательство Юрайт, 2025. — 108 с. — (Профессиональное образование). — ISBN 978-5-534-20429-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/563861>

Дополнительная литература:

1. Огнева, М. В. Программирование на языке С++: практический курс : учебник для среднего профессионального образования / М. В. Огнева, Е. В. Кудрина, А. А. Казачкова. — 2-е изд., перераб. и доп. — Москва : Издательство

Юрайт, 2026. — 342 с. — (Профессиональное образование). — ISBN 978-5-534-18975-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/585905>

2. Федоров, Д. Ю. Программирование на Python : учебник для среднего профессионального образования / Д. Ю. Федоров. — 6-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 187 с. — (Профессиональное образование). — ISBN 978-5-534-19654-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/585807>

3. Черпаков, И. В. Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 196 с. — (Профессиональное образование). — ISBN 978-5-534-18760-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/584552>

Интернет-ресурсы:

<http://inform-school.narod.ru> - Изучаем алгоритмизацию
<http://www.mathprog.narod.ru> - Математика и программирование
<http://www.firststeps.ru> - Первые шаги: уроки программирования
<http://ips.ifmo.ru> - Российская интернет-школа информатики и программирования
<http://www.codenet.ru> - CodeNet — все для программиста
<http://www.gosuslugi.ru/> - Государственные электронные услуги
<http://mon.gov.ru/> (Сайт Министерства образования и науки РФ) / www.edu.ru (Российский образовательный портал).
www.ed.gov.ru (Сайт Федерального агентства по образованию РФ).
Российская государственная публичная библиотека <http://elibrary.rsl.ru/>
<http://www.rambler.ru> – Русская поисковая система.
<http://www.yandex.ru> – Русская поисковая система.
<http://biblioteka.net.ru> – Библиотека компьютерных учебников.
Портал «Всеобуч»- справочно-информационный образовательный сайт, единое окно доступа к образовательным ресурсам –<http://www.edu-all.ru/>

Рекомендации по составлению и заполнению глоссария:

Использование глоссария способствует прояснению, запоминанию и уточнению ключевых понятий дисциплины. Глоссарий охватывает все узкоспециализированные термины, встречающиеся в тексте. Глоссарий должен содержать не менее 30 терминов, они должны быть перечислены в алфавитном порядке, соблюдена нумерация. Глоссарий должен быть оформлен по принципу реферативной работы, в обязательном порядке присутствует титульный лист и нумерация страниц. Объем работы должен составлять 10-15 страниц. Тщательно

проработанный глоссарий помогает избежать разночтений и улучшить в целом качество всей документации. В глоссарии включаются самые частотные термины и фразы, а также все ключевые термины с толкованием их смысла. Глоссарии могут содержать отдельные слова, фразы, аббревиатуры, слоганы и даже целые предложения.

Рекомендации по поиску и анализу источников и литературы

Найдите дополнительные источники литературы и источники, максимально соответствующие теме (заданию) и рекомендованные преподавателем. Ознакомьтесь с аннотацией источника, ее содержанием по оглавлению. Найдите необходимый материал и просмотрите его содержание. Определите форму записи (конспект, тезисы, цитаты, выписки, факты), составьте план и обсудите с преподавателем. Выпишите незнакомые слова и найдите их определение в словаре или спросите у преподавателя. Выполните краткий конспект. В конце своих записей сделайте резюме к тексту.

Рекомендации по подготовке сообщений, презентаций, докладов, рефератов

Внимательно прочитайте тему доклада, составьте предполагаемый план раскрытия темы, обсудите его с преподавателем. При работе с источниками информации тщательно отбирайте примеры, факты, следите за логикой их обоснования. Помните, что объем вашего выступления не должен выходить за предложенные преподавателем нормы. Выберите из подготовленного текста наиболее существенные моменты будущего выступления и согласуйте с преподавателем. При необходимости сопроводите свое выступление иллюстрациями или подготовьте презентацию. Презентация должна быть краткой, доступной и целостной. Главное в презентации – содержание. Придерживайтесь единого стилевого оформления. Не рекомендуется использовать в стилевом оформлении презентации более трех цветов или трех типов шрифта. Оформление слайда не должно отвлекать от его содержательной части. Не все слайды презентации должны быть выдержаны в одном стиле.

На одном слайде не должно быть слишком много информационных блоков, обычно до трёх. Ключевые слова в информационном блоке необходимо выделить. Наиболее важную информацию – поместить в центр слайда. Приводите факты, цифры, графики - это хорошая поддержка для вашего выступления. Голый текст никого не заинтересует (если вы, конечно, не несете слушателям свет новой истины). Применяйте высококонтрастные цвета, крупные шрифты и четкие иллюстрации. Фотографии, рисунки и другие иллюстрации старайтесь размещать

на отдельных слайдах. То же относится большим диаграммам, таблицам, схемам и графикам. Не забывайте указывать источники информации.

Приступая к подготовке реферата, доклада, на первом этапе, нужно собрать информацию. Можно использовать книги, статьи, научные публикации и интернет-ресурсы. Во время исследования нужно делать заметки, выделяя ключевые моменты, факты и примеры. Доклад необходимо структурировать. Обычно он включает три раздела: Введение - представить тему и объяснить её актуальность. Основная часть - разделить на несколько логически связанных частей. Каждая часть должна содержать ключевые идеи и факты. Заключение - подвести итоги, сделать выводы, дать рекомендации или предложить направления для дальнейшего изучения темы.

Подготовить визуальные материалы. Если доклад предполагает наличие презентации, нужно создать слайды, которые помогут визуализировать информацию.

Рекомендации по выполнению практических работ

Ознакомьтесь с темой, целью и заданием практической работы. Повторите теоретический материал, необходимый для выполнения работы.

Пройдите инструктаж по технике безопасности, если работа связана с оборудованием или лабораторными условиями. Запишите дату, тему и цель работы.

Следуйте предложенному алгоритму действий, если он есть. Работайте аккуратно, используйте линейки, карандаши, разборчивый почерк. Таблицы и схемы оформляйте чётко и по правилам.

По итогам работы сформулируйте выводы, ответьте на контрольные вопросы.

Проверьте, чтобы все расчёты и ответы были выполнены полностью и правильно.

Если работа письменная, составьте план, продумайте логику изложения, аргументируйте свои решения.

При возникновении трудностей обращайтесь к преподавателю или дополнительной литературе.

Соблюдайте требования к оформлению: аккуратность, читаемость, соответствие стандартам.

Рекомендации по самостоятельному изучению:

Начните с базовых понятий каждой категории

Решайте практические задачи регулярно

Используйте несколько источников для лучшего понимания

Создавайте конспекты и схемы

Применяйте изученное на практике

Участвуйте в онлайн-форумах и обсуждениях
Проверяйте свои знания через тесты и контрольные работы

Дополнительные советы:

Регулярность - занимайтесь минимум 30 минут ежедневно

Системность - следуйте логической последовательности изучения

Практика - решайте как можно больше задач

Обратная связь - проверяйте решения и исправляйте ошибки

Мотивация - ставьте конкретные цели и отслеживайте прогресс

Задания по самостоятельной работе должны быть выполнены до начала промежуточной аттестации.

Критерии оценки защиты мультимедийной презентации:

Оценка 5 ставится, если выполнены все требования к созданию мультимедийной презентации: обозначена тема, выполнен краткий анализ различных источников по данной теме, сформулированы выводы, выдержан объём, соблюдены требования к оформлению: фон, шрифт, объём текста, рисунки, анимация, выдержан единый стиль оформления

Оценка 4 – основные требования созданию мультимедийной презентации и ее защите выполнены, но при этом допущены недочёты: имеются неточности в изложении материала; отсутствует логическая последовательность; не выдержан объём презентации; имеются упущения в оформлении: не полностью соблюден единый стиль оформления.

Оценка 3 – имеются существенные отступления от требований к созданию мультимедийной презентации: не полностью раскрыта тема, имеются существенные недочеты в оформлении.

Оценка 2 – тема презентации не раскрыта, не соблюден единый стиль оформления слайдов.

Критерии оценки выступления с сообщением:

Оценка 5 - выполнены все требования к написанию и защите сообщения: обозначена проблема и обоснована её актуальность, сделан краткий анализ различных точек зрения на рассматриваемую проблему и логично изложена собственная позиция, сформулированы выводы, тема раскрыта полностью, выдержан объём, соблюдены требования к внешнему оформлению, даны правильные ответы на дополнительные вопросы.

Оценка 4 – основные требования к сообщению и их защите выполнены, но при этом допущены недочёты. В частности, имеются неточности в изложении материала; отсутствует логическая последовательность в суждениях; не выдержан объём; имеются упущения в оформлении; на дополнительные вопросы при защите даны неполные ответы.

Оценка 3 – имеются существенные отступления от требований: тема освещена лишь частично; допущены фактические ошибки в содержании или при ответе на дополнительные вопросы; во время защиты отсутствует вывод.

Оценка 2 – тема не раскрыта, обнаруживается существенное непонимание проблемы.

Критерии оценки устных ответов:

Оценка 5 – обучающийся полно излагает изученный материал, дает правильное определение языковых понятий; обнаруживает понимание материала, может обосновать свои суждения, применить знания на практике, привести необходимые примеры не только по учебнику, но и самостоятельно составленные; излагает материал последовательно и правильно с точки зрения норм литературного языка.

Оценка 4 – обучающийся дает ответ, удовлетворяющий тем же требованиям, что и для оценки «5», но допускает 1-2 ошибки, которые сам же исправляет, и 1-2 недочета в последовательности и языковом оформлении излагаемого.

Оценка 3 – обучающийся обнаруживает знание и понимание основных положений данной темы, но излагает материал неполно и допускает неточности в определении понятий и формулировке правил; не умеет достаточно глубоко и доказательно обосновать свои суждения и привести свои примеры; излагает материал непоследовательно и допускает ошибки в языковом оформлении излагаемого.

Оценка 2 – обучающийся обнаруживает незнание большей части соответствующего раздела изучаемого материала, допускает ошибки в формулировке определений и правил, искажающие их смысл, беспорядочно и неуверенно излагает материал. Оценка «2» отмечает такие недостатки в подготовке обучающегося, которые являются серьезным препятствием к успешному овладению последующим материалом.

5.1.2. Задания для повторения и углубления приобретаемых знаний.

№	Задания
---	---------

1	Понятие алгоритма и его свойства (дискретность, определённости, массовость, результативность). Способы описания алгоритмов (словесный, графический, псевдокод)
2	Понятие языка программирования и его классификация. Структура программы и правила её построения
3	Типы данных и их характеристики. Операторы и выражения
4	Линейные структуры (массивы, строки). Нелинейные структуры (списки, стеки, очереди)
5	Подпрограммы и функции. Параметры и их передача
6	Принципы ООП (инкапсуляция, наследование, полиморфизм). Классы и объекты
7	Основные алгоритмы сортировки (быстрая сортировка, сортировка слиянием, пирамидальная сортировка) Алгоритмы поиска (линейный, бинарный, хеш-поиск)
8	Типы файлов и режимы доступа Операции файлового ввода-вывода
9	Виды ошибок в программах Методы отладки
10	Функциональное программирование Параллельное и распределённое программирование

1.2. Задания, направленные на формирование профессиональных умений.

№	Задания
11	Прогнозировать поведение программы на основе анализа кода
12	Описывать алгоритмы различными способами (блок-схемы, псевдокод)
13	Работать с различными типами данных и структурами
14	Реализовывать основные операции над структурами
15	Создавать и использовать подпрограммы
16	Проектировать классы и их взаимосвязи
17	Тестировать программное обеспечение
18	Создавать и модифицировать файлы различных типов

19	Решать задачи обработки числовой информации
20	Применять принципы чистого кода

6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ПРОВЕДЕНИЯ ТЕКУЩЕГО КОНТРОЛЯ И ПРОМЕЖУТОЧНОЙ АТТЕСТАЦИИ ОБУЧАЮЩИХСЯ ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

6.1. Средства оценивания в ходе текущего контроля:

Перечень знаний, осваиваемых в рамках учебной дисциплины

Результаты обучения	Критерии оценки	Методы оценки
- понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции. - разрабатывать алгоритмы для конкретных задач	- знание и понимание материала; - грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	- устные опросы в ходе аудиторных занятий на знание категорий дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; контрольная работа; - экзамен.
- эволюцию языков программирования, их классификацию, понятие	- знание и понимание материала;	- устные опросы в ходе аудиторных занятий на знание категорий

системы программирования. - использовать программы для графического отображения алгоритмов. Определять сложность работы алгоритмов	- грамотное использование терминологии в ходе устных и письменных ответов; - критическое мышление и решение проблем; - коммуникативные навыки;	дисциплины; - письменные опросы в ходе аудиторных занятий на знание категорий дисциплины; - сообщения или доклады в ходе практических занятий; - выполнение заданий в ходе практических занятий: - выполнение заданий и упражнений, рекомендованных для самостоятельной работы; контрольная работа; - экзамен.
--	---	---

6.2. ФОС для текущего контроля:

См. Приложение 3 «ФОС для текущего контроля»

6.3 ФОС для промежуточной аттестации:

Задания для оценки знаний.

№	Задания
1	Вопросы к экзамену 1-20
2	1. Структура программы в C++. 2. Какие слова являются зарезервированными в C++? 3. Главное меню. 4. Каковы возможности и в чем преимущества интегрированной среды программирования? 5. Перечислите основные файлы среды программирования C++ и их назначение. Как запустить среду программирования C++? 6. Перечислите основные компоненты окна редактирования программ среды программирования C++. В чем их назначение?

	7. Опишите значение информации в строке статуса окна редактирования интегрированной среды программирования: a. F1 Help; F2 Save; F3 Open; Alt+F9 Compile; F9 Make; Alt+F10 Local menu) 8. Как откомпилировать файл программы? 9. В чем отличие пункта Run от пункта Compile главного меню интегрированной среды программирования? 10. Как посмотреть результаты выполнения программы в окне пользователя? 11. Что такое переменная? 12. В каком порядке выполняются строки программы? 13. Что такое программа линейной структуры? 14. Типы данных в C++. 15. Интерпретация блоков при графическом способе записи алгоритма. 16. Что такое оператор? Чем отличаются простые и структурные операторы? 17. Оператор безусловного перехода, его назначение и особенности применения. 18. Что представляет собой составной оператор? Как ограничиваются операторы, объединенные в составной оператор? 19. Назначение, формы записи и порядок выполнения оператора условия if. 20. Особенности использования вложенных условных операторов.
3	Вопросы к экзамену 21-40
4	21. Каковы отличия оператора выбора case от оператора условия if? 22. Какие правила должны выполняться при использовании оператора выбора case? 23. Что такое Циклы? 24. Какая базовая структура соответствует циклу “До” и “Пока”? 25. Формат строки цикла с параметрами? 26. Когда используется оператор цикла с параметрами? 27. Формат строки цикла с предусловием? 28. Формат строки цикла с послесловием? 29. Что называется подпрограммой? В чем состоит

	сходство и различие подпрограмм-процедур и подпрограмм-функций в языке С++? 30. В чем различие между стандартными и определенными пользователем подпрограммами? Приведите примеры. 31. Запишите синтаксическую диаграмму определения процедуры, функции. 32. Опишите последовательность событий при вызове процедуры или функции. 33. Что называется параметром, и каково его назначение? Формальные, фактические параметры, их взаимосвязь. 34. Каковы отличия параметров-значений от параметров-переменных, особенности их описания и применения. 35. Каковы особенности параметров-процедур и параметров-функций? 36. Чем отличаются локальные и глобальные параметры? Какова область их действия? 37. В каких случаях требуется предварительное или внешнее описание подпрограмм? Каковы особенности использования подпрограмм с предварительным описанием? 38. Что такое массив? 39. Как определить местоположение элемента в массиве? 40. Что такое индекс?
5	Вопросы к экзамену 41-60
6	41. Особенности расположения элементов массива в памяти ЭВМ? 42. Как объявить в программе массив (одномерный, двумерный). 43. Общие и отличительные черты одномерных, двумерных и n – мерных массивов? 44. Какие действия можно производить над массивом целиком и при каких условиях? 45. Как обратиться к элементу массива? 46. Какие операции можно производить над элементами массива? 47. Какими способами можно ввести (вывести) элементы массива? 48. Почему при описании массива предпочтительнее употреблять константы, а не указывать размеры массива в явном виде? 49. Какие способы сортировки массивов вы знаете?

	50. Что такое строка? 51. Каким идентификатором определяются данные строкового типа? 52. Какова максимально возможная длина строки? Как определить текущую длину строки? 53. Какие выражения называют строковыми? 54. Какие операции допустимы над строковыми данными? 55. Каким образом производится сравнение строк? 56. Как можно обратиться к отдельным символам строки? 57. Как производится сцепление строк? 58. Назначение специальных процедур и функций обработки данных строкового типа. Привести примеры. 59. Что такое запись?
7	Вопросы к экзамену 61-80
8	61. Отличие записи от массива? 62. Что из себя представляют элементы записи? 63. Формат записи? 64. Для чего нужен оператор With? 65. Что такое файл? 66. Для каких целей применяются файлы? 67. Характерные особенности файлов? 68. Какие устройства компьютера Т.Р. использует наряду с файлами? Каковы логические имена этих устройств? 69. Зачем используется специальная файловая переменная? 70. Какие процедуры служат для открытия файлов? Их общие черты и отличие? 71. Затем применяется процедура Close? 72. Для каких целей используется специальная функция IOResult? 73. Дать определение текстового файла? Из чего состоит содержимое текстового файла? 74. Назначение процедуры Append Назначение и отличительные особенности процедур Read и Write от Readln и Writeln. 75. Какие файлы сохраняются на диске после трансляции программы; 76. Что позволяет выполнить встроенный в ТР отладчик; 77. Перечислить способы выполнения программы до определенной

	точки. Раздел 4. Принципы объектно-ориентированного программирования 78. Что такое ООП? 79. Какая основная цель ОО-методологии? 80. В чем состоит отличие типа «Объект» от типа «Запись».
9	Вопросы к экзамену 81-90
10	81. Что называется объектом, свойством, методом, классом? 82. Что такое инкапсуляция? 83. Что означает A:= NEW(A)? 84. Что такое память? Что такое параграф? 85. Что такое адрес и из чего он состоит? 86. Что такое «Куча»? 87. Что дает директива \$M? 88. Что такое указатель? 89. Особенности объявления данных динамической структуры. 90. Процедуры GetMem, FreeMem

Задания для оценки умений.

См. Приложение 4 «Задания для самостоятельной работы»

7. ПЕРЕЧЕНЬ ОСНОВНОЙ И ДОПОЛНИТЕЛЬНОЙ УЧЕБНОЙ ЛИТЕРАТУРЫ, НЕОБХОДИМОЙ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ (МОДУЛЯ)

Основная литература:

1. Кудрина, Е. В. Основы алгоритмизации и программирования на языке C#: учебник для среднего профессионального образования / Е. В. Кудрина, М. В. Огнева. — Москва : Издательство Юрайт, 2026. — 322 с. — (Профессиональное образование). — ISBN 978-5-534-10772-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/587050>

2. Паронджанов, В. Д. Алгоритмические языки и программирование: ДРАКОН: учебник для вузов / В. Д. Паронджанов. — Москва : Издательство Юрайт, 2026. — 436 с. — (Высшее образование). — ISBN 978-5-534-13146-8. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/588170>

3. Трофимов, В. В. Основы алгоритмизации и программирования: учебник для среднего профессионального образования / В. В. Трофимов,

Т. А. Павловская. — 4-е изд. — Москва : Издательство Юрайт, 2025. — 108 с. — (Профессиональное образование). — ISBN 978-5-534-20429-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/563861>

Дополнительная литература:

1. Огнева, М. В. Программирование на языке C++: практический курс : учебник для среднего профессионального образования / М. В. Огнева, Е. В. Кудрина, А. А. Казачкова. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 342 с. — (Профессиональное образование). — ISBN 978-5-534-18975-9. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/585905>

2. Федоров, Д. Ю. Программирование на Python : учебник для среднего профессионального образования / Д. Ю. Федоров. — 6-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 187 с. — (Профессиональное образование). — ISBN 978-5-534-19654-2. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/585807>

3. Черпаков, И. В. Основы программирования : учебник и практикум для среднего профессионального образования / И. В. Черпаков. — 2-е изд., перераб. и доп. — Москва : Издательство Юрайт, 2026. — 196 с. — (Профессиональное образование). — ISBN 978-5-534-18760-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/584552>

Интернет-ресурсы:

<https://urait.ru> Образовательная платформа Юрайт. Для вузов и ссузов
<http://inform-school.narod.ru> - Изучаем алгоритмизацию
<http://www.mathprog.narod.ru> - Математика и программирование
<http://www.firststeps.ru> - Первые шаги: уроки программирования
<http://ips.ifmo.ru> - Российская интернет-школа информатики и программирования
<http://www.codenet.ru> - CodeNet — все для программиста
<http://www.gosuslugi.ru/> - Государственные электронные услуги
<http://mon.gov.ru/> (Сайт Министерства образования и науки РФ) / www.edu.ru (Российский образовательный портал).
www.ed.gov.ru (Сайт Федерального агентства по образованию РФ).
Российская государственная публичная библиотека <http://elibrary.rsl.ru/>
<http://www.rambler.ru> – Русская поисковая система.
<http://www.yandex.ru> – Русская поисковая система.
<http://biblioteka.net.ru> – Библиотека компьютерных учебников.

Портал «Всеобуч»- справочно-информационный образовательный сайт, единое окно доступа к образовательным ресурсам –<http://www.edu-all.ru/>

8. ПЕРЕЧЕНЬ КОМПЛЕКТОВ ЛИЦЕНЗИОННОГО И СВОБОДНО РАСПРОСТРАНЯЕМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, ИСПОЛЪЗУЕМОГО ПРИ ИЗУЧЕНИИ УЧЕБНОЙ ДИСЦИПЛИНЫ

При изучении учебной дисциплины (в том числе в интерактивной форме) предполагается применение современных информационных технологий. Комплект программного обеспечения для их использования включает в себя: свободно распространяемое программное обеспечение OpenOffice; веб-браузер (Google Chrome); электронные библиотечные системы IPR Smart и ЮРАЙТ; систему размещения в сети «Интернет» и проверки на наличие заимствований курсовых, научных и выпускных квалификационных работ «Антиплагиат.ВУЗ».

Для доступа к учебному плану и результатам освоения дисциплины, формирования Портфолио обучающегося используется Личный кабинет студента (он-лайн доступ через сеть Интернет <https://portal.rosnou.ru>). Для обеспечения доступа обучающихся во внеучебное время к электронным образовательным ресурсам учебной дисциплины ЮРАЙТ и IPR Smart, а также для студентов, обучающихся с применением дистанционных образовательных технологий, используется портал через сеть Интернет <https://e-edu.rosnou.ru>, <https://id.rosnou.ru>.

9. ПЕРЕЧЕНЬ РЕСУРСОВ ИНФОРМАЦИОННО-ТЕЛЕКОММУНИКАЦИОННОЙ СЕТИ «ИНТЕРНЕТ», НЕОБХОДИМЫХ ДЛЯ ОСВОЕНИЯ ДИСЦИПЛИНЫ

<https://urait.ru> Образовательная платформа Юрайт. Для вузов и ссузов
<http://inform-school.narod.ru> - Изучаем алгоритмизацию
<http://www.mathprog.narod.ru> - Математика и программирование
<http://www.firststeps.ru> - Первые шаги: уроки программирования
<http://ips.ifmo.ru> - Российская интернет-школа информатики и программирования
<http://www.codenet.ru> - CodeNet — все для программиста
<http://www.gosuslugi.ru/> - Государственные электронные услуги
<http://mon.gov.ru/> (Сайт Министерства образования и науки РФ) / www.edu.ru (Российский образовательный портал).
www.ed.gov.ru (Сайт Федерального агентства по образованию РФ).
Российская государственная публичная библиотека <http://elibrary.rsl.ru/>
<http://www.rambler.ru> – Русская поисковая система.
<http://www.yandex.ru> – Русская поисковая система.
<http://biblioteka.net.ru> – Библиотека компьютерных учебников.

Портал «Всеобуч»- справочно-информационный образовательный сайт, единое окно доступа к образовательным ресурсам –<http://www.edu-all.ru/>

10. ОБУЧЕНИЕ ИНВАЛИДОВ И ЛИЦ С ОГРАНИЧЕННЫМИ ВОЗМОЖНОСТЯМИ ЗДОРОВЬЯ

Изучение учебной дисциплины обучающимися инвалидами и лицами с ограниченными возможностями здоровья осуществляется в соответствии с Приказом Министерства просвещения РФ от 31 марта 2025 г. № 253 «Об утверждении Порядка обеспечения условий доступности для инвалидов объектов и предоставляемых услуг в сфере общего, среднего профессионального образования и соответствующего дополнительного профессионального образования, профессионального обучения, дополнительного образования детей и взрослых, организации отдыха и оздоровления детей, а также оказания им при этом необходимой помощи», Положением АНО ВО «РосНОУ» Об организации обучения студентов – инвалидов и лиц с ограниченными возможностями здоровья.

Лица с ограниченными возможностями здоровья и инвалиды обеспечиваются электронными образовательными ресурсами, адаптированными к состоянию их здоровья.

Предоставление специальных технических средств обучения коллективного и индивидуального пользования, подбор и разработка учебных материалов для обучающихся с ограниченными возможностями здоровья производится преподавателями с учетом индивидуальных психофизиологических особенностей обучающихся и специфики приема-передачи учебной информации на основании просьбы, выраженной в письменной форме.

С обучающимися по индивидуальному плану или индивидуальному графику проводятся индивидуальные занятия и консультации.

11. ОПИСАНИЕ МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЙ БАЗЫ, НЕОБХОДИМОЙ ДЛЯ ОСУЩЕСТВЛЕНИЯ ОБРАЗОВАТЕЛЬНОГО ПРОЦЕССА ПО ДИСЦИПЛИНЕ (МОДУЛЮ)

Учебная аудитория для проведения лекционных занятий (поточная), оснащенная оборудованием: специализированная мебель; маркерная доска (переносная), проектор (портативный); экран (переносной); ноутбук для преподавателя (с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду вуза); колонки; микрофон, наглядные пособия (плакаты).

Учебная аудитория для проведения практических занятий, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, воспитательной работы (**Лаборатория информационных технологий,**

программирования и баз данных), оснащенная оборудованием: специализированная мебель; магнитно-маркерная доска; проектор; экран; ПК для преподавателя (с возможностью подключения к сети "Интернет" и обеспечением доступа в электронную информационно-образовательную среду вуза), персональные компьютеры в сборке с пакетом специализированных программ; наглядные пособия (плакаты).

Помещение для самостоятельной работы обучающихся, оснащенное оборудованием: персональные компьютеры в сборке с пакетом специализированных программ, с выходом в интернет и обеспечением доступа в электронную информационно-образовательную среду.

Учебная аудитория для занятий обучающихся из числа инвалидов и лиц с ОВЗ. Специализированная мебель (стол для инвалидов-колясочников СИ-1), проектор; экран; мнемосхема тактильная; персональные компьютеры в сборке со специализированным пакетом программ, клавиатура с подсветкой с возможностью работы слабовидящим; комплект для маркировки Брайлем (клавиатура для компьютера); индукционная система переносная ИЦР-2.

Контрольная работа

Задание № 1. Укажите, сколько раз выполнится тело цикла с данным заголовком:

```
i=1
while i<=10:
    i=i+1
```

Задание № 2. Что будет выведено на экран монитора в результате выполнения программы:

```
main.py
1  z=5
2  z1=z**3
3  print(z1)
4
```

Задание № 3. Укажите номер строки программы, в которой может возникнуть ошибка? Гарантируется, что на вход программы подаются только целые числа.

```
1  n = int(input())
2  k = int(input())
3  for i in range(n, k):
4      if i % 10 == 0:
5          t += 1
6      else:
7          t = 0
```

Задание № 4. Укажите, что будет выведено на экран при выполнении программы?

```
a = 1
b = 1
rez = 2
for i in range(2, 20, 3):
    a, b = b, a + b
    rez += 2 * b
print(rez)
```

Задание № 5. Дан построчный фрагмент программного кода, который принимает три положительных числа и определяет вид треугольника, длины сторон которого равны введенным числам. Дополните программу недостающим оператором в строке кода с номером 1.

№ строки	Программный код
1	
2	if a1 == a2 == a3:
3	print('Равносторонний')
4	elif a2 == a3 or a1 == a3 or a1 == a2:
5	print('Равнобедренный')
6	else:
7	print('Разносторонний')

Задание №6. Дан фрагмент программного кода, который принимает целое число x и определяет, принадлежит ли данное число промежутку от -1 до 17. Дополните оператор if в строке с номером 2 необходимым условием.

№ строки	Программный код
1	a1 = int(input())
2	if _____:
3	print('Принадлежит')
4	else:
5	print('Не принадлежит')

Вопросы к экзамену

1. Структура программы в C++.
2. Какие слова являются зарезервированными в C++?
3. Главное меню.
4. Каковы возможности и в чем преимущества интегрированной среды программирования?
5. Перечислите основные файлы среды программирования C++ и их назначение. Как запустить среду программирования C++?
6. Перечислите основные компоненты окна редактирования программ среды программирования C++. В чем их назначение?
7. Опишите значение информации в строке статуса окна редактирования интегрированной среды программирования:
F1 Help; F2 Save; F3 Open; Alt+F9 Compile; F9 Make; Alt+F10 Local menu)
8. Как откомпилировать файл программы?
9. В чем отличие пункта Run от пункта Compile главного меню интегрированной среды программирования?
10. Как посмотреть результаты выполнения программы в окне пользователя?
11. Что такое переменная?
12. В каком порядке выполняются строки программы?
13. Что такое программа линейной структуры?
14. Типы данных в C++.
15. Интерпретация блоков при графическом способе записи алгоритма.
16. Что такое оператор? Чем отличаются простые и структурные операторы?
17. Оператор безусловного перехода, его назначение и особенности применения.
18. Что представляет собой составной оператор? Как ограничиваются операторы, объединенные составной оператор?
19. Назначение, формы записи и порядок выполнения оператора условия if.
20. Особенности использования вложенных условных операторов.
21. Каковы отличия оператора выбора case от оператора условия if?
22. Какие правила должны выполняться при использовании оператора выбора case?
23. Что такое Циклы?
24. Какая базовая структура соответствует циклу “До” и “Пока”?
25. Формат строки цикла с параметрами?
26. Когда используется оператор цикла с параметрами?
27. Формат строки цикла с предусловием?

28. Формат строки цикла с послесловием?
29. Что называется подпрограммой? В чем состоит сходство и различие подпрограмм-процедур и подпрограмм-функций в языке C++?
30. В чем различие между стандартными и определенными пользователем подпрограммами? Приведите примеры.
31. Запишите синтаксическую диаграмму определения процедуры, функции.
32. Опишите последовательность событий при вызове процедуры или функции.
33. Что называется параметром, и каково его назначение? Формальные, фактические параметры, их взаимосвязь.
34. Каковы отличия параметров-значений от параметров-переменных, особенности их описания и применения.
35. Каковы особенности параметров-процедур и параметров-функций.
36. Чем отличаются локальные и глобальные параметры? Какова область их действия?
37. В каких случаях требуется предварительное или внешнее описание подпрограмм? Каковы особенности использования подпрограмм с предварительным описанием?
38. Что такое массив?
39. Как определить местоположение элемента в массиве?
40. Что такое индекс?
41. Особенности расположения элементов массива в памяти ЭВМ?
42. Как объявить в программе массив (одномерный, двумерный).
43. Общие и отличительные черты одномерных, двумерных и n – мерных массивов?
44. Какие действия можно производить над массивом целиком и при каких условиях?
45. Как обратиться к элементу массива?
46. Какие операции можно производить над элементами массива?
47. Какими способами можно ввести (вывести) элементы массива?
48. Почему при описании массива предпочтительнее употреблять константы, а не указывать размеры массива в явном виде?
49. Какие способы сортировки массивов вы знаете?
50. Что такое строка?
51. Каким идентификатором определяются данные строкового типа?
52. Какова максимально возможная длина строки? Как определить текущую длину строки?
53. Какие выражения называют строковыми?
54. Какие операции допустимы над строковыми данными?
55. Каким образом производится сравнение строк?
56. Как можно обратиться к отдельным символам строки?

57. Как производится сцепление строк?
58. Назначение специальных процедур и функций обработки данных
59. строкового типа. Привести примеры.
60. Что такое запись?
61. Отличие записи от массива?
62. Что из себя представляют элементы записи?
63. Формат записи?
64. Для чего нужен оператор With?
65. Что такое файл?
66. Для каких целей применяются файлы?
67. Характерные особенности файлов?
68. Какие устройства компьютера Т.Р. использует наряду с файлами? Каковы логические имена этих устройств?
69. Зачем используется специальная файловая переменная?
70. Какие процедуры служат для открытия файлов? Их общие черты и отличие?
71. Затем применяется процедура Close?
72. Для каких целей используется специальная функция IOResult?
73. Дать определение текстового файла? Из чего состоит содержимое текстового файла?
74. Назначение процедуры Append Назначение и отличительные особенности процедур Read и Write от Readln и Writeln.
75. Какие файлы сохраняются на диске после трансляции программы;
76. Что позволяет выполнить встроенный в ТР отладчик;
77. Перечислить способы выполнения программы до определенной точки.
78. Что такое ООП?
79. Какая основная цель ОО-методологии?
80. В чем состоит отличие типа «Объект» от типа «Запись».
81. Что называется объектом, свойством, методом, классом?
82. Что такое инкапсуляция?
83. Что означает A:= NEW(A)?
84. Что такое память? Что такое параграф?
85. Что такое адрес и из чего он состоит?
86. Что такое «Куча»?
87. Что дает директива \$M?
88. Что такое указатель?
89. Особенности объявления данных динамической структуры.
90. Процедуры GetMem, FreeMem.

Практические работы

Практическое занятие № 1 Разработка алгоритма словесным способом, формульно–словесным, блок – схемным, псевдокодом, структурной диаграммой

Пример: При выполнении условия $x > 0$ вычисляется функция: $z = x + y$, иначе, а именно, когда $x = 0$ или $x < 0$, вычисляется функция: $z = x^2 + y^2$.

Составить блок-схему вычисления функции по алгоритму ветвления. Значения переменных x, y могут быть любые, вводить их с клавиатуры.

Решение: На рис.2 представлен разветвляющийся алгоритм, где в зависимости от условия выполнится одна из веток. В блок-схеме появился новый блок 3, который проверяет условие задачи. Остальные блоки знакомы из линейного алгоритма.

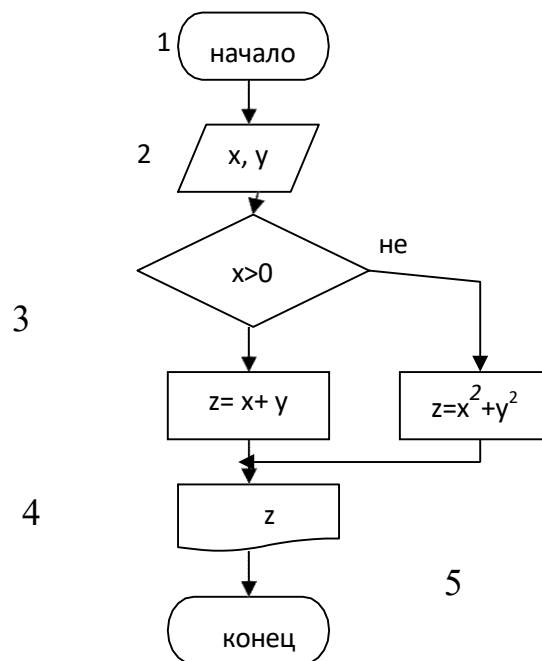


Рис.2. Алгоритм ветвления

Пример: Найти максимальное значение из трёх различных целых чисел, введенных с клавиатуры. Составить блок-схему решения задачи.

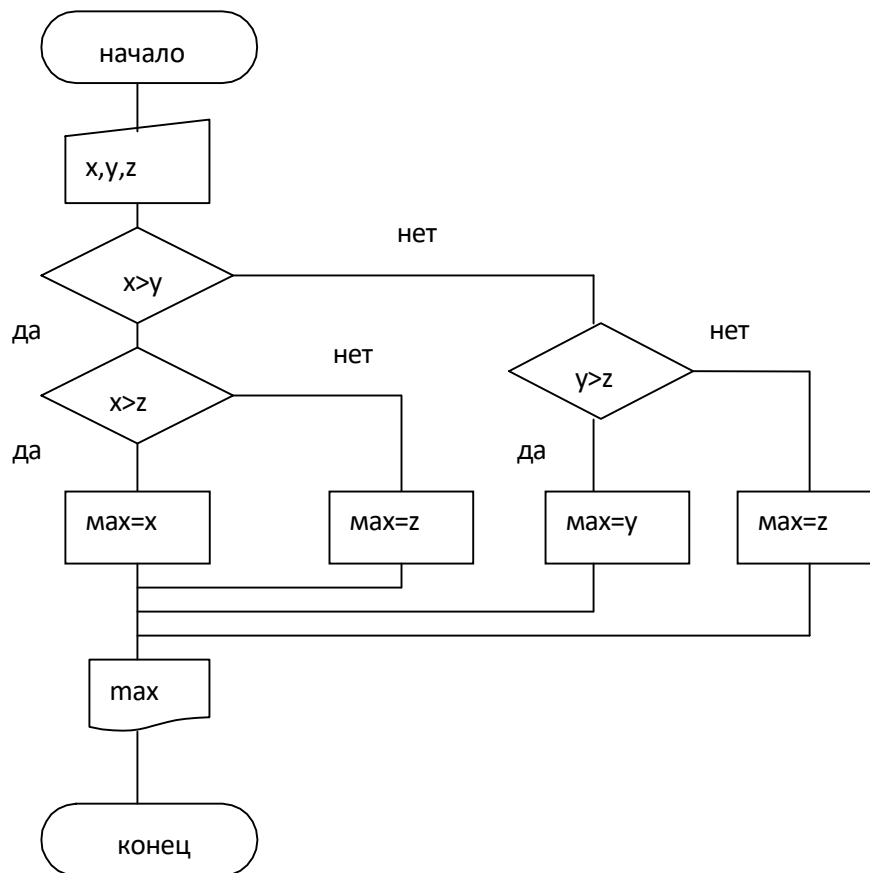


Рис. 3. Блок-схема поиска максимума

Решение: Данный алгоритм предполагает проверку условия. Для этого выбирается любая из трёх переменных и сравнивается с другими двумя. Если она больше, то поиск максимального числа окончен. Если условие не выполняется, то сравниваются две оставшиеся переменные. Одна из них будет максимальной. Блок-схема к этой задаче представлена на рис 3.

Циклические алгоритмы

Циклический алгоритм предусматривает повторение одной операции или нескольких операций в зависимости от условия задачи.

Из циклических алгоритмов выделяют два типа:

- 1) с заданным количеством циклов или со счётчиком циклов;
- 2) количество циклов неизвестно.

Пример: В цикле вычислить значение функции $z = x * y$ при условии, что одна из переменных x меняется в каждом цикле на единицу, а другая переменная y не меняется и может быть любым целым числом. В результате выполнения цикла при начальном значении переменной $x = 1$ можно получить

таблицу умножения. Количество циклов может быть любым. Составить блок-схему решения задачи.

Решение: В примере количество циклов задаётся. Соответственно выбирается алгоритм циклов первого типа. Алгоритм этой задачи приводится на рис. 4.

Во втором блоке вводятся количество циклов n и любые целые числа x, y .

В блок-схеме появился новый блок 3, в котором переменная i считает количество циклов, после каждого цикла увеличиваясь на единицу, пока счётчик не будет равен $i=n$. При $i=n$ будет выполнен последний цикл.

В третьем блоке указывается диапазон изменения счётчика цикла (от $i=1$ до $i=n$).

В четвёртом блоке изменяются значения переменных: z, x .

В пятом блоке выводится результат. Четвёртый и пятый блоки повторяются в каждом цикле.

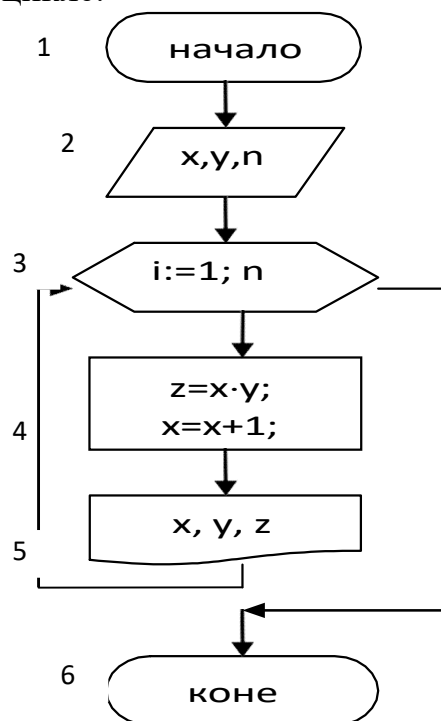


Рис.4 . Циклический алгоритм со счётчиком циклов

Этот тип циклических алгоритмов предпочтителен, если дано количеством циклов.

Если количество циклов неизвестно, то блок-схемы циклических алгоритмов могут быть представлены в виде рисунков 5, 6.

Пример: Вычислить $y=y-x$ пока $y>x$, если $y=30$, $x=4$. Подсчитать количество выполненных циклов, конечное значение переменной y . В цикле вывести значение переменной y , количество выполненных циклов. Составить

блок-схему решения задачи.

Решение: В примере количество циклов неизвестно. Соответственно выбирается алгоритм циклов второго типа. Алгоритм этой задачи приводится на рис. 5.

Условие проверяется на входе в цикл. В теле цикла выполняется два блока:

- 1) $y = y - x; i = i + 1$;
- 2) вывод значений переменных i, y .

Цикл выполняется до тех пор, пока выполняется условие $y > x$. При условии равенства этих переменных $y = x$ или $y < x$ цикл заканчивается.

Алгоритм, представленный на рис.5, называется *циклический алгоритм с предусловием*, так как условие проверяется в начале цикла или на входе в цикл.

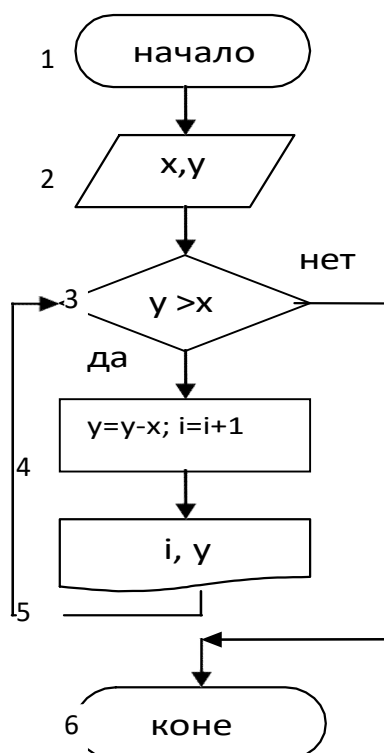


Рис.5. Блок-схема циклического алгоритма с предусловием

Во втором блоке вводятся $y=30, x=4$.

В третьем блоке проверяется условие $y > x$ на входе в цикл. Если условие выполняется, то переход к блоку 4, иначе на блок 6.

В четвёртом блоке вычисляется значение переменной y , подсчитывается количество выполненных циклов $i = i + 1$.

В пятом блоке выводится результат:

- значение переменной y ,
- количество выполненных циклов i .

Пример: Составить блок-схему примера (рисунок 5), проверяя условие

выхода из цикла. В этом примере условие задачи не меняется, и результат выведется тот же, но блок-схема будет другой.

Решение: В этом случае проверяется условие на выход из цикла: $y \leq x$. При этом условии цикл не выполняется. Условие в блок-схеме следует перенести в конец цикла, после вывода на печать. Цикл выполняется до тех пор, пока выполняется условие $y > x$.

Алгоритм, если условие перенести в конец цикла, называется *алгоритмом цикла с постусловием*.

Алгоритм этой задачи приводится на рис. 6.

Во втором блоке вводятся $y=30, x=4$.

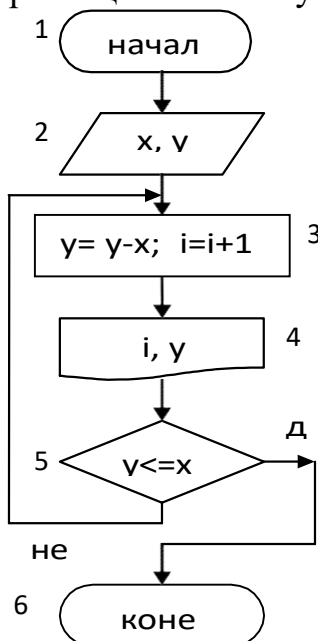
В третьем блоке вычисляется значение переменной y , подсчитывается количество выполненных циклов $i=i+1$.

В четвёртом блоке выводится результат:

- значение переменной y ,
- количество выполненных циклов i .

В пятом блоке проверяется условие $y \leq x$ на выход из цикла. Если условие выполняется, то переход к блоку 6, иначе на блок 3 и цикл повторяется.

Рис.6 . Алгоритм цикла с постусловием



1. При заданном X условие выполняется? Написать результат вычисления и ответ попадаем в условие или нет.

Входные данные по вариантам

№	X1	X1
1	55	12

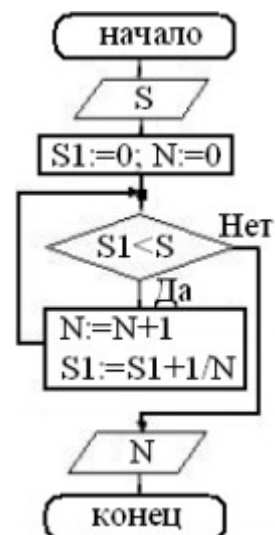


2	85	13
3	24	17
4	65	15
5	17	54
6	15	67
7	26	3
8	27	21
9	92	34
10	12	23
11	45	22
12	66	45
13	71	46
14	13	76

2. Написать результат выполнения алгоритма с указанными входными данными

Входные данные по вариантам

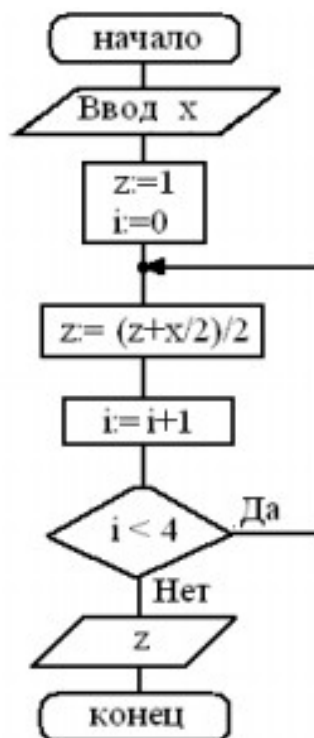
№	S
1	1,5
2	1,8
3	2,4
4	1,6
5	1,7
6	1,3
7	2,6
8	2,37
9	1,92
10	1,12
11	1,45
12	2,66
13	2,71
14	2,13
15	1,45



3. Написать результат выполнения алгоритма с указанными входными данными

Входные данные по вариантам

№	X
1	-1
2	0
3	1
4	2
5	3
6	4
7	5
8	6
9	7
10	-3
11	-4
12	-5
13	-6
14	-7
15	7



4. Построить блок схему к задаче(по вариантам). Указать тип алгоритма, что дано и что нужно найти.

№	Задача
1	Дано двузначное число. Найти: число десятков в нем;
2	Дано двузначное число. Найти: число единиц в нем;
3	Дано двузначное число. Найти: сумму его цифр;
4	Дано двузначное число. Найти: произведение его цифр.
5	Дано двузначное число. Получить число, образованное при перестановке цифр заданного числа.
6	Дано трехзначное число. Найти: число единиц в нем;
7	Дано трехзначное число. Найти: число десятков в нем;
8	Дано трехзначное число. Найти: сумму его цифр;
9	Дано трехзначное число. Найти: произведение его цифр.
10	Дано трехзначное число. Найти число, полученное при прочтении его цифр справа налево.
11	Дано трехзначное число. В нем зачеркнули первую слева цифру и приписали ее в конце. Найти полученное число.
12	Дано трехзначное число. В нем зачеркнули последнюю справа цифру и приписали ее в начале. Найти полученное число.
13	Дано трехзначное число. Найти число, полученное при перестановке первой и второй цифр заданного числа
14	Дано трехзначное число. Найти число, полученное при перестановке второй и третьей цифр заданного числа.

15	Дано трехзначное число, в котором все цифры различны. Получить шесть чисел, образованных при перестановке цифр заданного числа.
----	---

5. Построить блок схему к задаче (по вариантам). Указать тип алгоритма, что дано и что нужно найти.

№	Задача
1	Определить максимальное и минимальное значения из двух различных вещественных чисел.
2	Известны два расстояния: одно в километрах, другое — в футах (1 фут 0,305 м). Какое из расстояний меньше?
3	Известны две скорости: одна в километрах в час, другая — в метрах в секунду. Какая из скоростей больше?
4	Даны радиус круга и сторона квадрата. У какой фигуры площадь больше?
5	Даны объемы и массы двух тел из разных материалов. Материал какого из тел имеет большую плотность?
6	Известны сопротивления двух несоединенных друг с другом участков электрической цепи и напряжение на каждом из них. По какому участку протекает меньший ток?
7	Даны вещественные числа a, b, c ($a \neq 0$). Выяснить, имеет ли квадратное уравнение с данными параметрами решение
8	Известны площади круга и квадрата. Определить: уместится ли круг в квадрате?
9	Известны площади круга и квадрата. Определить: уместится ли квадрат в круге?
10	Известны площади круга и равностороннего треугольника. Определить: уместится ли круг в треугольнике?
11	Известны площади круга и равностороннего треугольника. Определить: уместится ли треугольник в круге?
12	Дано двузначное число. Определить: какая из его цифр больше: первая или вторая;
13	Дано двузначное число. Определить: одинаковы ли его цифры
14	Дано двузначное число. Определить: кратна ли трем сумма его цифр;
15	Дано двузначное число. Определить: кратна ли сумма его цифр числу a .

Практическая работа 2. Разработка алгоритмов линейной и разветвляющейся структуры

1 Теоретические сведения

Алгоритм — это последовательность команд, предназначенная исполнителю, в

результате выполнения которой он должен решить поставленную задачу.

Дискретность — алгоритм должен представлять процесс решения задачи как последовательное выполнение некоторых простых шагов.

Детерминированность (определённость) – каждая команда должна быть однозначно определена.

Понятность — алгоритм должен включать только те команды, которые доступны исполнителю и входят в его систему команд.

Завершаемость (конечность) — при корректно заданных исходных данных алгоритм должен завершать работу и выдавать результат за конечное число шагов.

Массовость (универсальность). Алгоритм должен быть применим к разным наборам исходных данных.

Результативность — завершение алгоритма определёнными результатами.

Безошибочность – это если алгоритм даёт правильные результаты для любых допустимых исходных данных.

Под переменной будем понимать некоторую ячейку памяти, т. е. отдельное место для хранения одной константы.

Формульно-словесный способ представления алгоритмов основан на задании инструкций о выполнении конкретных действий в четкой последовательности в сочетании со словесными пояснениями.

На алгоритмическом языке. Алгоритмический язык – совокупность правил и обозначений, используемые для записи алгоритма.

Графический способ (метод блок-схемы). При таком представлении алгоритма, каждый этап отображается в виде геометрических фигур-блоков, форма которых зависит от выполняемой операции.

Вычислительное действие или последовательность действий	
Ввод-вывод в общем виде	
Начало, конец алгоритма	
Проверка условий	

Линейный алгоритм – это алгоритм, в котором блоки выполняются последовательно сверху вниз от начала до конца.

Любая переменная имеет имя – идентификатор. По правилам языков программирования имя переменной должно начинаться с буквы и может содержать буквы (только латинские), цифры и знак подчеркивания.

Арифметические действия и выражения:

«+» - сложение;

«-» - вычитание;

«*» - умножение;

«/» - деление;

mod - нахождение остатка от деления;

div - деление нацело (находить остаток от деления и делить нацело можно только целые числа);

для указания порядка действий используются только круглые скобки, их может быть несколько, главное, чтобы количество открывающихся скобок равнялось количеству закрывающихся;

sqr(x) – возведение аргумента в квадрат;

sqrt(x) – извлечение квадратного корня;

abs(x) – модуль.

Для типов переменной слева и арифметического выражения справа от знака присваивания выполняются правила:

если переменная вещественного типа, то арифметическое выражение может быть как целого, так и вещественного типа, выражение преобразуется к вещественному типу;

если переменная слева целого типа, то арифметическое выражение только целочисленное.

Разветвляющимся называется такой алгоритм, в котором выбирается один из нескольких возможных вариантов вычислительного процесса. Каждый подобный путь называется ветвью алгоритма.

Признаком разветвляющегося алгоритма является наличие операций проверки условия. Различают два вида условий - простые и составные.

Простым условием (отношением) называется выражение, составленное из двух арифметических выражений или двух текстовых величин (иначе их еще называют операндами), связанных одним из знаков:

< - меньше, чем...

> - больше, чем...

<= - меньше, чем... или равно

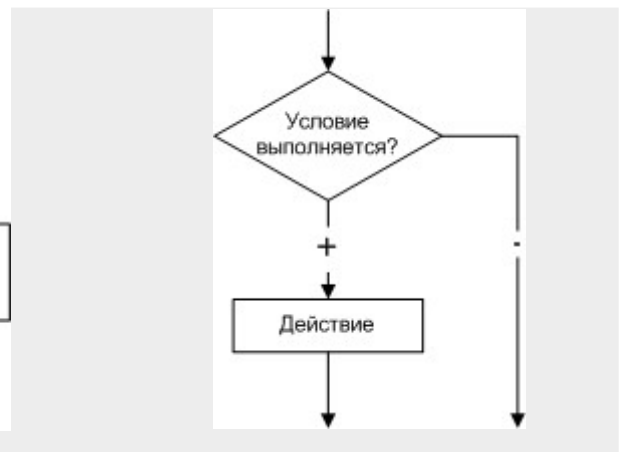
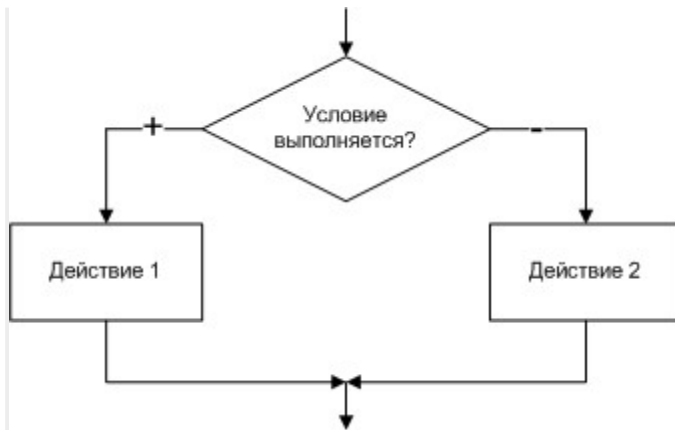
>= - больше, чем... или равно

<> - не равно

□ = - равно

Составное условие - объединение нескольких условий в одну группу.

В блок-схемах разветвленные алгоритмы изображаются так, как показано на рисунках



2 Задания

Задание 1 Написать линейный алгоритм решения задачи в словесно-формульном виде и с помощью блок-схемы.

Задание 2 Написать разветвляющий алгоритм решения задачи в словесно-формульном виде и с помощью блок-схемы.

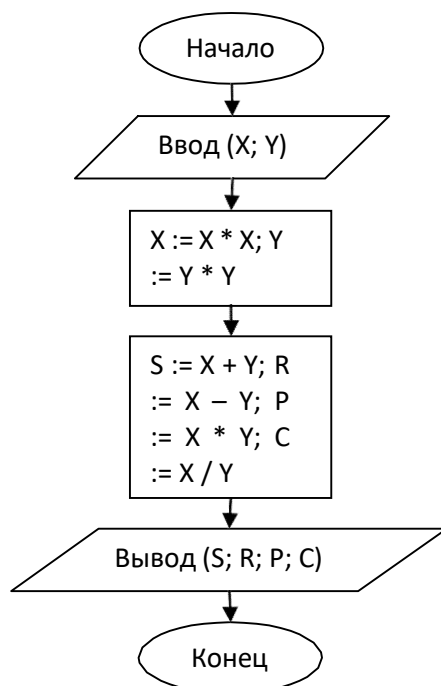
Задание 1

Исходные данные:

Даны два ненулевых числа. Найти сумму, разность, произведение и частное этих чисел

Решение:

1 Алгоритм в графическом и словесно-формульном виде:



1. Начало;
2. Ввод (X; Y);
3. $X := X * X$;
4. $Y := Y * Y$;
5. $S := X + Y$;
6. $R := X - Y$;
7. $P := X * Y$;
8. $C := X / Y$;

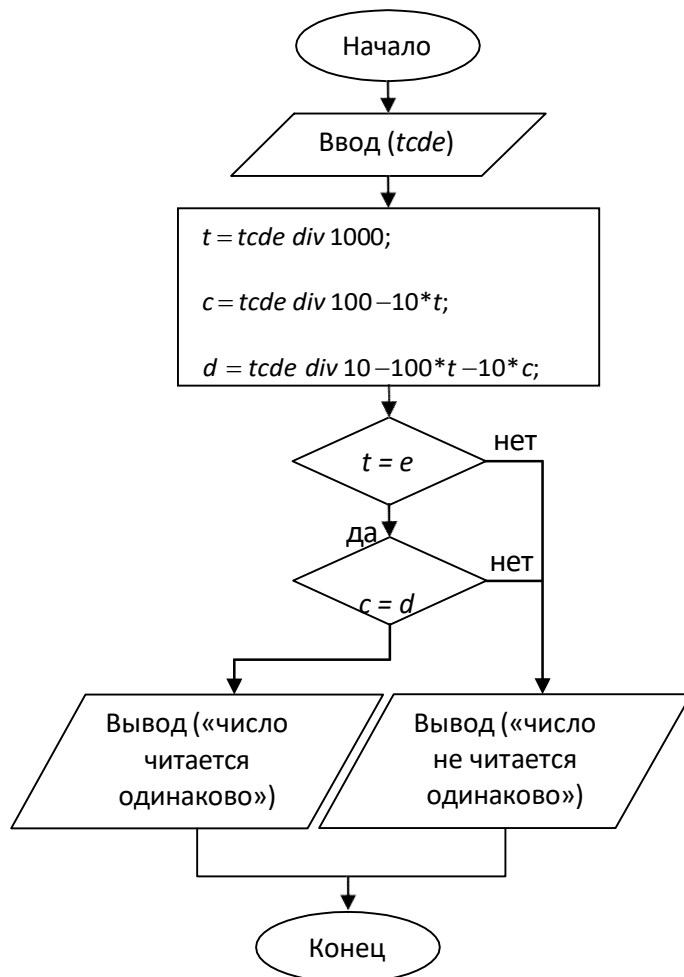
9. Вывод (S; R; P; C);
10. Конец.

Задание 2

1 Составить и записать алгоритм решения задачи в графическом и словесно-формульном виде.

Исходные данные:

Дано четырехзначное число. Проверить истинность высказывания: «Данное число читается одинаково слева направо и справа налево».



1. Начало;
2. Ввод ($tcde$);
3. $t \leftarrow tcde \div 1000$;
4. $c \leftarrow tcde \div 100 - 10 * t$;
5. $d \leftarrow tcde \div 10 - 100 * t - 10 * c$;
6. $e \leftarrow tcde \div 1000 * t - 100 * c - 10 * d$
7. Если $t = e$ то
 Если $c = d$ то
 Вывод («число читается одинаково»);
 Иначе вывод («число не читается одинаково»);
8. Конец.

Варианты к практической работе:

Задание 1

- 1 Даны длины ребер a , b , c прямоугольного параллелепипеда. Найти его объем и площадь поверхности
- 2 Дана длина ребра куба a . Найти его объем и площадь поверхности
- 3 Даны катеты прямоугольного треугольника a и b . Найти его гипотенузу c , площадь S и периметр P
- 4 Найти значение функции $y = 3x^6 - 6x^2 - 7$ при данном значении x .
- 5 Найти значение функции $y = 4(x-3)^6 - 7(x-3)^3 + 2$ при данном значении x .
- 6 Найти длину окружности L и площадь круга S заданного радиуса R : В качестве значения π использовать 3.14.
- 7 Даны переменные A , B , C . Изменить их значения, переместив содержимое A в C , C — в B , B — в A , и вывести новые значения переменных A , B , C .
- 8 Известно, что X кг шоколадных конфет стоит A рублей, а Y кг ирисок стоит B рублей. Определить, сколько стоит 1 кг шоколадных конфет, 1 кг ирисок, а также во сколько раз шоколадные конфеты дороже ирисок.
- 9 Поменять местами содержимое переменных A и B и вывести новые значения A и B .
- 10 Даны переменные A , B , C . Изменить их значения, переместив содержимое A в B , B — в C , C — в A , и вывести новые значения переменных A , B , C .
- 11 Основание прямой призмы — прямоугольный треугольник с катетами a и b . Высота призмы равна h . Найти площадь поверхности и объем призмы.
- 12 Сторона основания правильной четырехугольной пирамиды равна a , а высота — h . Найти площадь поверхности и объем пирамиды.
- 13 Найти объем шара радиуса R и площадь его поверхности.
- 14 Найти площадь поверхности и объем кругового цилиндра, радиус основания которого равен R , а высота h .
- 15 Найти площадь осевого сечения прямого кругового конуса с радиусом основания R и образующей L .
- 16 Найти площадь поверхности и объем прямого кругового конуса с радиусом основания R и высотой h .
- 17 Сторона основания правильной треугольной пирамиды равна a , а высота — h . Найти площадь поверхности и объем пирамиды.
- 18 Сторона основания правильной шестиугольной пирамиды равна a , а высота — h . Найти площадь поверхности и объем пирамиды.
- 19 Сторона основания правильной треугольной призмы равна a . Высота призмы равна h . Найти площадь поверхности и объем призмы.
- 20 Сторона основания правильной шестиугольной призмы равна a . Высота призмы равна h . Найти площадь поверхности и объем призмы.
- 21 Даны длины ребер a , b , c прямоугольного параллелепипеда. Найти

длину диагонали параллелепипеда

22 Найти площадь поверхности прямого кругового усеченного конуса с радиусом оснований R_1 , R_2 и высотой h .

23 Найти объем прямого кругового усеченного конуса с радиусом оснований R_1 , R_2 и образующей L .

24 Координаты концов отрезка – (x_1, y_2) и (x_2, y_2) . Вычислить координаты точки, делящей отрезок пополам.

25 Найти объем шара, если площадь его поверхности равна S .

26 Найти площадь поверхности и объем кругового цилиндра, высота которого равна h , а площадь осевого сечения – S .

Задание 2

1 Даны числа x , y . Проверить истинность высказывания: «Точка с координатами (x, y) лежит во второй координатной четверти».

2 Даны числа x , y . Проверить истинность высказывания: «Точка с координатами (x, y) лежит в четвертой координатной четверти».

3 Даны числа x , y . Проверить истинность высказывания: «Точка с координатами (x, y) лежит во второй или третьей координатной четверти».

4 Даны числа x , y . Проверить истинность высказывания: «Точка с координатами (x, y) лежит в первой или третьей координатной четверти».

5 Даны два целых числа: A , B . Проверить истинность высказывания: «Справедливы неравенства $A > 2$ и $B < 3$ ».

6 Даны два целых числа: A , B . Проверить истинность высказывания: «Справедливы неравенства $A > 0$ или $B < -2$ ».

7 Даны три целых числа: A , B , C . Проверить истинность высказывания: «Справедливо двойное неравенство $A < B < C$ ».

8 Даны три целых числа: A , B , C . Проверить истинность высказывания: «Число B находится между числами A и C ».

9 Даны три целых числа: A , B , C . Проверить истинность высказывания: «Каждое из чисел A , B , C положительное».

10 Даны три целых числа: A , B , C . Проверить истинность высказывания: «Хотя бы одно из чисел A , B , C положительное».

11 Даны два числа. Если их значения не равны, то присвоить каждому сумму этих значений, а если равны, то присвоить числам нулевые значения. Вывести новые значения A и B .

12 Даны два числа. Если их значения не равны, то присвоить каждому большее из этих значений, а если равны, то присвоить числам нулевые значения. Вывести новые значения A и B .

13 Даны три числа. Найти наименьшее из них.

14 Даны три числа. Найти среднее из них (то есть число, расположенное

между наименьшим и наибольшим).

15 Даны три числа. Вывести вначале наименьшее, а затем наибольшее из данных чисел.

16 Даны два целых числа: A , B . Проверить истинность высказывания: «Справедливы неравенства $A > 10$ и $B < 0$ ».

17 Даны два целых числа: A , B . Проверить истинность высказывания: «Справедливы неравенства $A > 0$ и $B < -3$ ».

18 Даны три числа. Найти наибольшее из них.

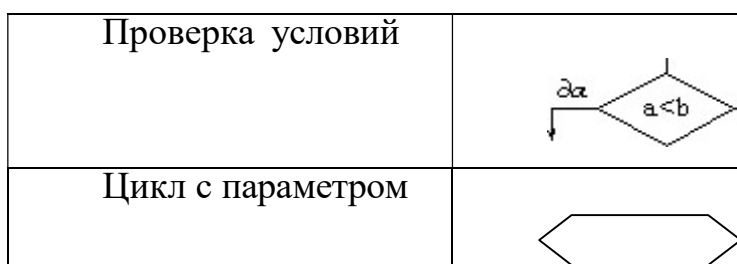
19 Даны два целых числа: A , B . Проверить истинность высказывания: «Справедливы неравенства $A > 5$ и $B < -5$ ».

20 Даны два целых числа: A , B . Проверить истинность высказывания: «Справедливы неравенства $A > 1$ и $B < -3$ ».

Практическая работа № 3. Разработка алгоритмов циклической структуры

Основные понятия.

Графический способ представления циклических алгоритмов (метод блок-схемы).



Алгоритм, в котором вычисления повторяются по одной и той же совокупности формул, называется циклическим

Существуют следующие конструкции для организации циклов:

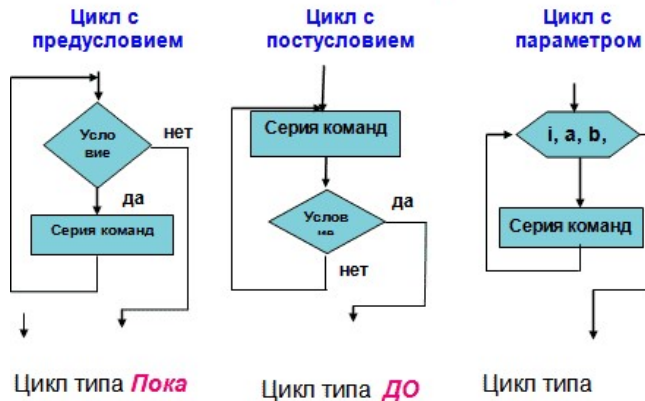
цикл с предусловием;

цикл с постусловием;

цикл со счетчиком.

Графическое представление циклических алгоритмов

Виды циклических алгоритмов



Циклы с предусловием используются тогда, когда выполнение цикла связано с некоторым логическим условием. Оператор цикла с предусловием имеет две части: условие выполнения цикла и тело цикла. При выполнении оператора цикла определенная группа операторов выполняется до тех пор, пока определенное в операторе условие истинно. Если условие сразу ложно, то оператор не выполнится ни разу.

Общая форма записи следующая Пока (условие)

нц

серия

команд кц

на Паскале:

while <булево

выражение> do begin

группа

операторов end;

Операторные скобки ставят, чтобы отделить от остальной программы ту группу операторов, которую нужно повторить в цикле. Если в цикле нужно выполнить только один оператор, то операторные скобки не ставят.

Циклы с постусловием используются тогда, когда выполнение цикла связано с некоторым логическим условием. Оператор цикла с предусловием имеет две

части: тело цикла и условие окончания выполнения цикла. При выполнении оператора цикла определенная группа операторов выполняется до тех пор, пока определенное в операторе условие не изменится. Оператор выполняется хотя бы один раз.

Задание

- 1 Составить и записать алгоритм решения задачи в графическом и словесно-формульном виде
- 2 Написать программу на языке Паскаль
- 3 Ввести программу и запустить её на исполнение.
- 4 Протестировать алгоритм с различными данными.
- 5 Отладить программу (найти и исправить возможные ошибки).
- 6 Проанализировать программу и сделать вывод.

```
program cikl;  
label m1;  
var I, N:integer;  
    S, Znam:real;  
begin  
    m1:  
    write('Введите целое положительное  
число '); readln(N);  
    if N<1 then  
    begin  
        writeln('Число введено неправильно');  
        goto m1;  
    end;  
    S:=0;  
    Znam:=1;  
    For I:=1 to N do
```

```

begin
    Znam:=Znam*I;
    S:=S+I/Znam;
end;
writeln('Результат = ', S);
end.

```

3 Программный код для запуска

```

program ciki;
label m1;
var I, N:integer;
    S, Znam:real;
begin
    m1:
        write('Введите целое положительное число ');
        readln(N);
        if N<1 then
            begin
                writeln('Число введено неправильно');
                goto m1;
            end;
        S:=0;
        Znam:=1;
        For I:=1 to N do
            begin
                Znam:=Znam*I;
                S:=S+I/Znam;
            end;
        writeln('Результат = ', S);
end.

```

4 Тестирование программы с различными

данными:

```

Введите целое положительное число 37
Результат = 2.71828182845905

Введите целое положительное число 0
Число введено неправильно
Введите целое положительное число 6
Результат = 2.71666666666667

```

5 Программа выдает верный результат, но формат числа неудобен для восприятия, достаточно оставить три знака после запятой в выводе результата, в коде программы необходимо добавить формат числа:

```

writeln('Результат = ', S:6:3);

```

```

Число введено неправильно
Введите целое положительное число 6
Результат = 2.71666666666667
Введите целое положительное число 36
Результат = 2.718
Введите четырехзначное число 0000
число читается одинаково

```

Задания к практической работе.

- 1 Даны действительные числа x , y . Вывести в порядке возрастания все целые числа, расположенные между x и y , а также количество этих чисел.
- 2 Даны действительные числа x , y . Вывести в порядке убывания все целые числа, расположенные между x и y , а также количество этих чисел.
- 3 Дано действительное число – цена 1 кг конфет. Вывести стоимость 1, 2 ... 15 кг конфет.
- 4 Дано действительное число – цена 1 кг конфет. Вывести стоимость 0,1; 0,2 ... 1,2 кг конфет.
- 5 Дано действительное число – цена 1 кг конфет. Вывести стоимость 0,5; 1; 1,5 ... 10 кг конфет.
- 6 Даны действительные числа x , y . Найти произведение всех целых чисел, расположенных между x и y , а также количество этих чисел.
- 7 Даны действительные числа x , y . Найти сумму квадратов всех целых чисел, расположенных между x и y , а также количество этих чисел.
- 8 Даны действительные числа x , y . Найти сумму кубов всех целых чисел, расположенных между x и y , а также количество этих чисел.
- 9 Проверить истинность высказывания: «Среди трех данных целых чисел есть хотя бы одна пара совпадающих».
- 10 Дано действительное число A и целое число N . Вывести все целые степени числа A от 1 до N .
- 11 Дано действительное число X и целое число N . Вывести N первых членов арифметической прогрессии и найти сумму ее первых членов, если первый член – X , а разность 1,12
- 12 Дано действительное число X и целое число N . Вывести N первых членов арифметической прогрессии и найти сумму ее первых членов, если первый член X , а разность $(-2,2)$
- 13 Дано целое число N . Последовательность чисел Фибоначчи F_K определяется следующим образом: $F_1 = 1$; $F_2 = 1$; $F_K = F_{K-2} + F_{K-1}$; $K = 3; 4; \dots$ Вывести элементы от F_1 до F_N
- 14 Дано действительное число X и целое число N . Вывести N первых членов геометрической прогрессии и найти сумму ее первых членов, если первый член X , а знаменатель 1,27
- 15 Дано действительное число X и целое число N . Вывести N первых членов геометрической прогрессии и найти сумму ее первых членов, если первый член X , а знаменатель $(-2,73)$
- 16 Дано действительное число X и целое число N . Вывести N первых членов геометрической прогрессии и найти сумму ее первых членов, если первый член X , а знаменатель 0,31

17 Дано действительное число X и целое число N . Вывести N первых членов арифметической прогрессии и найти сумму ее N первых членов, если первый член X , а разность $(-0,38)$.

Практическая работа №4. Составление блок-схем алгоритмов для обработки массивов

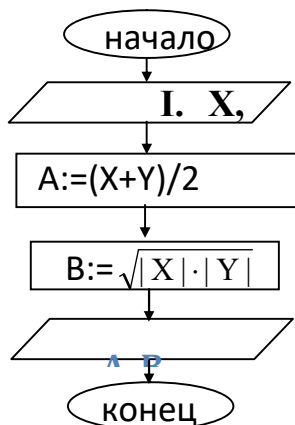
Пример 1 Даны 2 действительных числа. Требуется вычислить в программируемом режиме среднее арифметическое этих чисел и среднее геометрическое их модулей.

Решение:

```
10 REM "Вычисление ср.ар. и ср. геом."
20 INPUT X, Y
30 A=(X+Y)/2
40 B=SQR(ABS(X)*ABS(Y))
50 PRINT A,B
60 END
```

Контрольный счет:

X	Y	A	B
1	1	1	1
3	-5	-1	3.872983

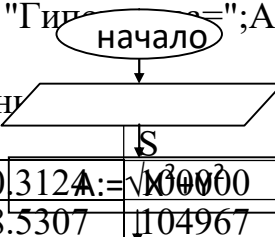


Пример 2 Даны катеты прямоугольного треугольника. Найти гипотенузу и площадь.

```
10 REM "Вычисление гипот. и площади"
20 INPUT "Введите значения катетов:";X,Y
30 A=SQR(X^2+Y^2)
40 S=X*Y/2
50 PRINT "Гипотенуза=";A;"Площадь=";S
60 END
```

Контрольный счет:

X	Y	A	S
400	500	640.3124	100000
321	654	728.5307	104967

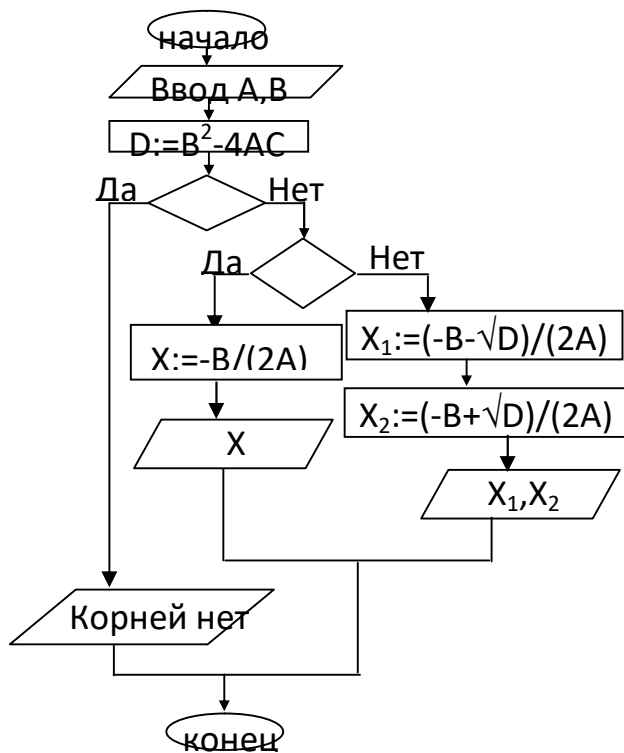


2. Разветвляющиеся программы.

Для реализации логического блока используется оператор условного перехода. Он изменяет ход выполнения программы от результата проверки условия. Эта управляющая конструкция называется развилка. На Бейсике развилка выражается с помощью оператора:

IF выражение THEN операторы ELSE операторы

Пример 3 Вычислить действительные корни квадратного уравнения $ax^2 + bx + c = 0$ при произвольных значениях коэффициентов a, b, c и условии $a \neq 0$.



```

10 REM "Вычисление корней"
20 INPUT A,B,C
30 D=B^2-4*A*C
40 IF D<0 THEN 50 ELSE 70
50 PRINT "Корней нет"
60 GOTO 140
70 IF D>0 THEN 80 ELSE 110
80 X=-B/(2*A)
90 PRINT X
100 GOTO 140
110 X1=(-B-SQR(D))/(2*A)
120 X2=(-B+SQR(D))/(2*A)
130 PRINT X1,X2
140 END
  
```

Контрольный счет:
при $n=5$ $n!=120$

Пример 4 Задана последовательность

$$a_n = \frac{1}{n^2 + 1}. \text{ Найти сумму } m \text{ первых членов}$$

последовательности.

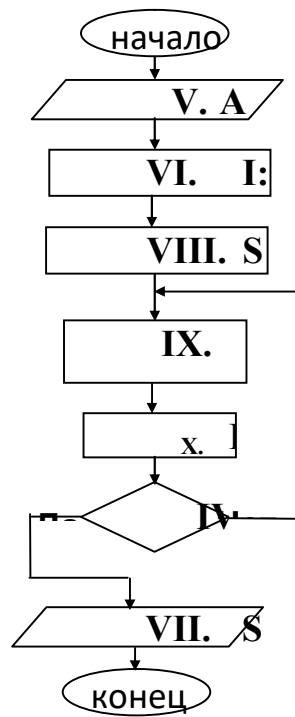
```

10 REM "НАХОЖДЕНИЕ СУММЫ"
20 INPUT M
30 N = 1
40 S = 0
50 A = 1 / (N ^ 2 + 1)
60 S = S + A
70 N = N + 1
80 IF N > M
THEN 90 ELSE
50
  
```

```

90 PRINT S
100 END
  
```

Контрольный счет:
 $m=13$ $S=1.002768$;
 $m=45$ $S=1.0547$



Пример 5 Дано действительное число a . Найти среди чисел $1, 1 + \frac{1}{2}, 1 + \frac{1}{2} + \frac{1}{3}$ и т.д. первое большее a .

```

10 REM "НАХОЖДЕНИЕ ЭЛЕМЕНТА"
20 INPUT A
30 I = 1
40 S = 0
50 S = S + 1/I
60 I = I + 1
70 IF S > A THEN 80 ELSE 50
80 PRINT S
90 END

```

Цикл с параметром реализуется с помощью операторов цикла со счетчиком:
 FOR счетчик = начало TO конец [STEP приращение]

блок операторов

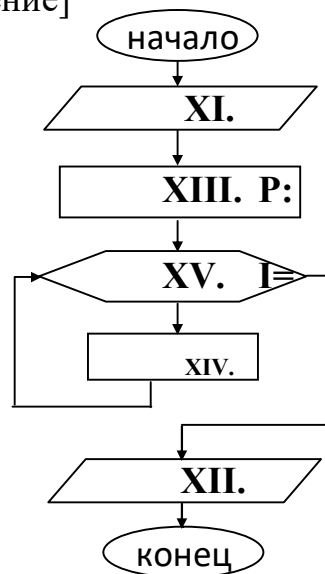
NEXT счетчик

Пример 6 Вычислить (для данного n) $n!$.

```

10 REM "Вычисление факториала"
20 INPUT N
30 P = 1
40 FOR I = 1 TO N
50 P = P * I
60 NEXT I
70 PRINT P
80 END

```



Практическое занятие №5. Анализ и сравнение синтаксисов

Задача. Движение автомобиля.

Что моделируется?	Процесс движения объекта «автомобиль»
Вид движения	Равноускоренное
Что известно о движении?	Начальная скорость (V_0), ускорение (a), макси-мально развиваемая скорость ($V_{\max}$)
Что надо найти?	Скорость (V_i) в заданные моменты времени (t_i).
Как задаются моменты времени?	От нуля через равные интервалы (Δt)?

Что ограничивает расчеты?	$V_i < V_{\max}$
---------------------------	------------------

Такие характеристики объекта, как цвет, тип кузова, год выпуска и общий пробег, степень изношенности шин и многие другие, в данной постановке учитывать не будем.

Разработка модели

Этап разработки модели начинается с построения информационной модели в различных знаковых формах, которые на завершающей стадии воплощаются в компьютерную модель. В информационных моделях задача приобретает вид, позволяющий принять решение о выборе программной среды и четко представить алгоритм построения компьютерной модели.

Информационная модель

Выбор наиболее существенных данных при формировании информационной модели и ее сложность определяются целью моделирования. Параметры объектов, определенных при формализации задачи, располагаются в порядке убывания значимости. При моделировании учитываются не все, а лишь некоторые свойства, интересующие исследователя.

Если отбросить существенные факторы, то модель будет неверно отражать оригинал (прототип). Если оставить их слишком много, модель окажется сложна для построения и исследования. Во многих исследованиях создают несколько моделей одного объекта, начиная от простейших, с минимальным набором определяющих параметров. Затем постепенно уточняют модель, добавляя некоторые из отброшенных характеристик.

Иногда задача может быть уже сформулирована в упрощенной форме, цель — четко поставлена, а параметры модели, которые надо учесть, — определены. Задачи такого вида вам приходилось неоднократно решать на уроках математики и физики. Однако в обычной жизни отбор информации приходится проводить самостоятельно.

Результатом построения информационной модели является хорошо знакомая вам таблица характеристик объекта. В зависимости от типа задачи таблица может видоизменяться.

Рассмотрим информационные модели описанной выше задачи.

Задача. Движение автомобиля

Объект моделирования	Параметры
Процесс движения автомобиля	V_0 – начальная скорость Δt – интервал изменения времени a – ускорение $V_{\max}$ – максимально развиваемая автомобилем скорость

t_i – время движения V_i – значения скорости

При построении компьютерной образно-знаковой модели (текстовый или графический документ) информационная модель будет описывать объекты, их параметры, а также предварительные исходные значения, которые исследователь определяет в соответствии со своим опытом и представлениями, а затем уточняет в ходе компьютерного эксперимента.

Информационная модель, как правило, представляется в той или иной знаковой форме.

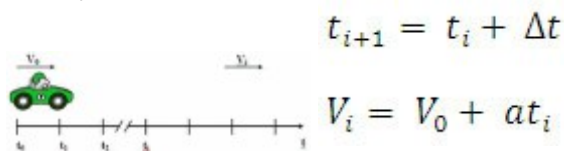
Таблица – один из примеров знаковых моделей.

Иногда полезно дополнить представление об объекте и другими знаковыми формами (схемой, чертежом, формулами), если это способствует лучшему пониманию задачи.

Рассмотрим знаковые модели для описанных выше задач.

Задача. Движения автомобиля.

Задача о движении автомобиля становится более понятной, если привести рисунок с указанием обозначений, используемых в задаче.



Математическая модель движения автомобиля имеет вид:

Правильно составленная математическая модель просто необходима в задачах, где требуется рассчитать значения параметров объекта.

Знаковые формы могут иметь и другой вид.

Например, при создании географических или исторических карт разрабатывается система условных обозначений.

И лишь для простых, знакомых по содержанию задач знаковые модели не требуются.

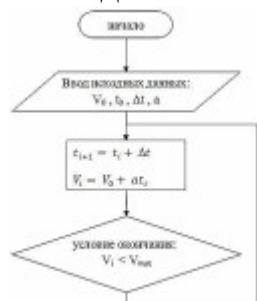
Компьютерная модель

Теперь, когда сформирована информационная знаковая модель можно приступать собственно к компьютерному моделированию - созданию компьютерной модели. Сразу возникает вопрос о средствах, которые необходимы для этого, то есть об инструментах моделирования.

Компьютерная модель - это модель, реализованная средствами программной среды. Существует множество программных комплексов, которые позволяют проводить построение и исследование моделей (моделирование). Каждая программная среда имеет свой инструментарий и позволяет работать с определенными видами информационных моделей. Поэтому перед исследователем возникает нелегкий вопрос выбора наиболее удобной и эффективной среды для решения поставленной задачи. Надо сказать, что одну и ту же задачу можно решить, используя различные среды.

От выбора программной среды зависит алгоритм построения компьютерной модели, а также форма его представления.

Например, это может быть блок-схема. На рисунке ниже представлен алгоритм задачи о движении автомобиля в виде блок-схемы. Руководствуясь блок-схемой, задачу можно решить в разных средах. В среде программирования - это программа, записанная на алгоритмическом языке. В прикладных средах это последовательность технологических приемов, приводящая к решению задачи.



Например, при моделировании в среде графического редактора или текстового процессора алгоритм может быть представлен в словесной форме, описывающей последовательность действий по созданию объектов и, если требуется, технологических приемов. При разработке алгоритма построения модели в электронных таблицах особое внимание обращается на выделение области исходных и расчетных данных и правила записи формул, связывающих данные разных областей.

Исходя из вышесказанного, можно сделать вывод, что при моделировании на компьютере необходимо иметь представление о классах программных средств, их назначении, инструментарии и технологических приемах работы. Разнообразное программное обеспечение позволяет преобразовать исходную информационную знаковую модель в компьютерную и провести компьютерный эксперимент.

Рассмотрим возможные варианты выбора компьютерной среды для приведенной выше задачи. Справедливости ради следует заметить, что предложенная в качестве иллюстрации задача может быть решена и зачастую решается без применения компьютера.

Задача. Движение автомобиля.

Для задач, в которых требуется получить расчетные значения, подходит среда электронных таблиц. В этой среде информационная и математическая модели объединяются в таблицу, содержащую три области: исходные данные, промежуточные расчеты и результаты. Электронная таблица позволяет не только рассчитать требуемые скорости, но и построить график движения автомобиля.

Выполнить тест

В таблице приведена стоимость перевозок между соседними железнодорожными станциями. Укажите схему, соответствующую таблице.

	A	B	C	D
--	---	---	---	---

A		4		5
B	4		3	6
C		3		
D	5	6		
1)	2)	3)	4)	

Путешественник пришел в 08:00 на автостанцию поселка ЛЕСНОЕ и увидел следующее расписание автобусов:

Отправление из **Прибытие в** **Время отправления** **Время прибытия**
ЛЕСНОЕ **ОЗЕРНОЕ** 07:45 08:55

ЛУГОВОЕ ЛЕСНОЕ 08:00 09:10

ПОЛЕВОЕ ЛЕСНОЕ 08:55 11:25

ПОЛЕВОЕ ЛУГОВОЕ 09:10 10:10

ЛЕСНОЕ ПОЛЕВОЕ 09:15 11:45

ОЗЕРНОЕ ПОЛЕВОЕ 09:15 10:30

ЛЕСНОЕ ЛУГОВОЕ 09:20 10:30

ОЗЕРНОЕ ЛЕСНОЕ 09:25 10:35

ЛУГОВОЕ ПОЛЕВОЕ 10:40 11:40

ПОЛЕВОЕ ОЗЕРНОЕ 10:45 12:00

Определите самое раннее время, когда путешественник сможет оказаться в пункте ПОЛЕВОЕ согласно этому расписанию.

1) 10:30 2) 11:25 3) 11:40 4) 11:45

Между населёнными пунктами А, В, С, D, E, F построены дороги, протяжённость которых приведена в таблице. (Отсутствие числа в таблице означает, что прямой дороги между пунктами нет.)

	A	B	C	D	E	F
A		5				
B	5		9	3	8	
C		9			4	
D		3			2	
E		8	4	2		7
F					7	

Определите длину кратчайшего пути между пунктами А и F (при условии, что передвигаться можно только по построенным дорогам).

1) 11 2) 13 3) 15 4) 17

Транспортная фирма осуществляет грузоперевозки разными видами транспорта между четырьмя городами: ЧЕРЕПОВЕЦ, МОСКВА, КУРСК, ПЕРМЬ. Стоимость доставки грузов и время в пути указаны в таблице:

Стоимость (у.е.)	E_i	Пункт назначения
------------------	-------	------------------

МОСКВА	ПЕРМЬ	100	70
МОСКВА	КУРСК	30	10

МОСКВА	ЧЕРЕПОВЕЦ	50	15
ПЕРМЬ	МОСКВА	100	69
ЧЕРЕПОВЕЦ	ПЕРМЬ	140	80
ЧЕРЕПОВЕЦ	МОСКВА	50	15
ЧЕРЕПОВЕЦ	КУРСК	100	80
КУРСК	ПЕРМЬ	60	40
КУРСК	МОСКВА	30	10
КУРСК	ЧЕРЕПОВЕЦ	100	80
КУРСК	ЧЕРЕПОВЕЦ	90	100

Определите маршрут наиболее дешевого варианта доставки груза из ЧЕРЕПОВЦА в ПЕРМЬ. Если таких маршрутов несколько, в ответе укажите наиболее выгодный по времени вариант.

ЧЕРЕПОВЕЦ – ПЕРМЬ

ЧЕРЕПОВЕЦ – КУРСК – ПЕРМЬ

ЧЕРЕПОВЕЦ – МОСКВА – ПЕРМЬ

ЧЕРЕПОВЕЦ – МОСКВА – КУРСК – ПЕРМЬ

В таблицах приведена стоимость перевозки грузов между соседними станциями. Если пересечение строки и столбца пусто, то соответствующие станции не являются соседними. Укажите номер таблицы, для которой выполняется условие «Максимальная стоимость перевозки грузов от пункта В до пункта D не больше 6».

1)					2)					3)					4)				
	A	B	C	D		A	B	C	D		A	B	C	D		A	B	C	D
A		2		2	A		2	1	1	A		1	3	6	A		3	2	1
B	2		4	3	B	2		4		B	1		2	4	B	3		2	
C		4		4	C	1	4		1	C	3	2			C	2	2		4
D	2	3	4		D	1		1		D	6	4			D	1		4	

Практическая работа №6. Знакомство со средой программирования

Краткие теоретические сведения

Основными операторами являются:

Read, Readln – оператор ввода



данных;

- := – оператор присваивания;
- Write, Writeln - оператор вывода.

Program - зарезервированное слово для обозначения заголовка программы. Var - зарезервированное слово для обозначения раздела переменных.

Begin и End - служебные слова, заключающие между собой раздел операторов.

Структура программы с линейной структурой

Окно программы Pascal ABC состоит из окна редактора кода, окна ввода и окна вывода.

Окно редактора кода

Большую часть рабочей области, её верхнюю часть занимает окно редактора кода. В него вводится исходный текст программы.

Горячие клавиши, которые можно использовать при работе с текстом программы: F2, Ctrl-S- сохранить файл.

F3, Ctrl-O- загрузить файл.

F12- сохранить файл под новым именем. Ctrl-Shift-S- сохранить все открытые файлы.

Ctrl-Tab, Ctrl-Shift-Tab – перейти к следующему/предыдущему окну редактора. Ctrl-Shift-I- увеличить отступ выделенного блока.

Ctrl-Shift-U – уменьшить отступ выделенного блока.

Под окном редактора расположено окно вывода. Оно предназначено для вывода данных процедурами write и writeln, а также для вывода сообщений об ошибках и предупреждений во время работы программы.

Окно вывода может быть скрыто. Клавиша F5 показывают/скрывают окно вывода. Для скрытия окна вывода используется также клавиша Esc.

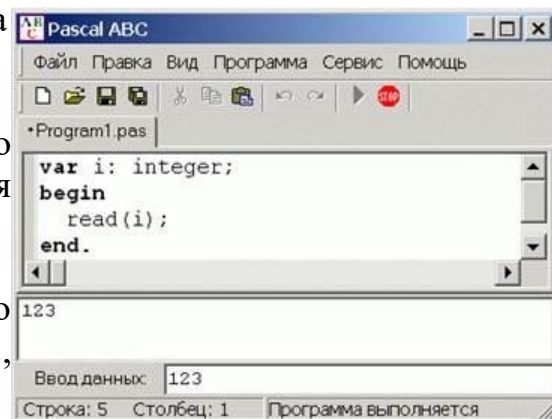
Окно вывода обязательно открывается при любом выводе в него.

Для очистки окна вывода следует нажать комбинацию клавиш Ctrl-Del или кнопку . Окно ввода открывается при выполнении процедур read и readln в ходе работы программы.

Ввод данных в окно ввода сопровождается эхо-выводом в окно вывода. После нажатия клавиши Enter данные из окна ввода попадают в соответствующие переменные, окно ввода закрывается, и программа продолжает работать дальше.

Запуск и остановка программы

Для запуска программы в текущем окне редактора следует нажать клавишу F9 или кнопку  панели инструментов.



Программа вначале компилируется во внутреннее представление, после чего, если не найдены ошибки, программа начинает выполняться. При выполнении программы кнопка за-пуска программы становится неактивной, кнопка останова программы, наоборот, активной и в строке статуса отображается информация “Программа выполняется”.

Выполнение программы можно в любой момент прервать нажатием комбинации клавиш Ctrl-F2 или кнопки . При этом в окне вывода появится сообщение: • Программа прервана пользователем.

Практические задания

Содержание отчета по результатам выполнения практической работы

Отчет должен содержать:

1. Название работы.
2. Цель работы.
3. Результат выполнения задания № 1 (записать программу на компьютер и в тетрадь).
4. Результат выполнения задания № 2 (записать программу на компьютер и в тетрадь)
5. Результат выполнения задания № 3 (записать программу на компьютер и в тетрадь)
6. Результат выполнения задания № 4 (записать программу на компьютер и в тетрадь)
7. Результаты выполнения задания № 5 (записать программу на компьютер и в тетрадь)
8. Результаты выполнения задания № 6 (ответить письменно на вопросы)
9. Вывод по работе (результат выполнения № 7).

Задание 1. Наберите текст программы, описывающей решение примера $C=A+B$. Выполните компиляцию программы и тестирование.

1. На рабочем столе запустите ярлык Pascal ABC.
2. В окне текстового редактора программы Pascal ABC наберите

текст программы: Program Z1; Var A, B, C: integer;

```
Begin
Writeln ('Задача 1');
Writeln ('Введите число A'); Readln (A);
Writeln ('Введите число B'); C:=A+B;
Writeln ('c=' , c ); Readln ;
```

End .

3. Проверьте программу на ошибки (компиляция) - нажать *Программа/Компилировать* (или Ctrl+F9).

4. Протестируйте программу:

1) запустить программу, выбрав в меню пункт *Программа/Выполнить* (или F9).

2) в ответ на приглашение «Введите число A» набрать 23,

3) в ответ на приглашение «Введите число B» набрать 17,

4) если программа работает правильно, то в результате будет выведено – C=40

5. Самостоятельно протестируйте программу с другими числовыми данными: A=2; B= -12;

A = -10; B= -17;

A = 0; B= - 20;

Запишите результаты в тетрадь.

Задание 2. Измените исходную программу, чтобы она находила **частное** двух чисел.

Задание 3. Измените исходную программу, чтобы она находила сумму четырех чисел.

Задание 4. Наберите текст программы, описывающей решение примера

Выполните компиляцию программы и тестирование.

$$D = \frac{6A + 2B}{4A}$$

1. В окне текстового редактора программы Pascal наберите текст программы: Program Z2;

Var A, B, D : real; Begin

Writeln ('Задача 2'); Writeln ('Введите A'); Readln (A);

Writeln ('Введите B'); Readln (B);

D:= (6*A+2*B)/ (4*A);

Writeln ('D= ', D); Readln ;

End.

2. Проверить программу на ошибки - нажать *Программа/Компилировать* (или Ctrl+F9).

3. Протестировать программу:

1) запустить программу, выбрав в меню пункт *Программа/Выполнить* (или F9)

2) в ответ на приглашение «Введите число A» набрать 2,

3) в ответ на приглашение «Введите число B» набрать 2,

4) если программа работает правильно, то в результате будет

выведено –

$E=2.0000000000E+0000$.

4. Введите ограничение для вещественного числа, для этого внесите изменения в текст программы `Writeln ('D = ', D:4:2);`

5. Самостоятельно протестируйте программу с другими числовыми данными: $A=2$; $B=1,5$;

$A = 3$; $B = -7$;

$A = 0$; $B = -2$;

Запишите результаты в тетрадь.

Задание 5. Создайте и протестируйте программу вычисления переменной $D=(a*b)+2$

Практическая работа №7. Составление программ линейной структуры

Цель:

освоить методы разработки алгоритмов и программирования задач линейной структуры.

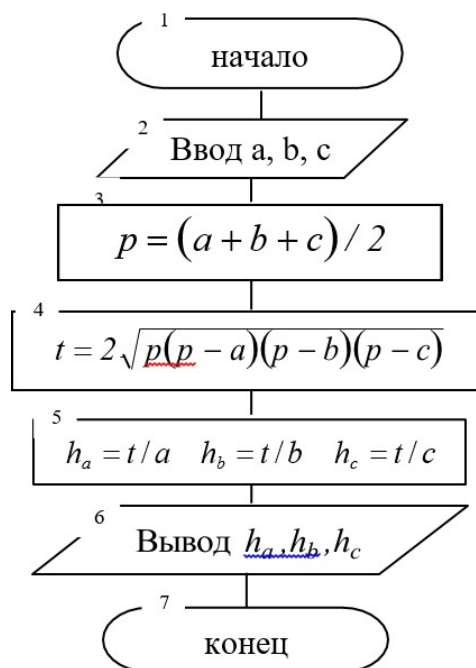
Краткие теоретические сведения

Алгоритм линейной структуры – это алгоритм, в котором все действия выполняются последовательно друг за другом и только один раз. Блок-схема алгоритма представляет собой последовательность блоков, которые располагаются сверху вниз в порядке их выполнения. Все промежуточные или исходные данные влияния на направление процесса выполнения не оказывают.

Пример. Вычислить высоты треугольника со сторонами a , b , c , используя формулу Герона.

При решении данной задачи для исключения повторений следует вычислять высоты не по приведенным выше формулам непосредственно, а используя промежуточную переменную:

Схема алгоритма решения задачи имеет следующий вид:



Для записи программы линейной структуры необходимы операторы присваивания, ввода исходных данных и вывода результатов вычислений.

Программа на языке Pascal состоит из заголовка, раздела описаний и раздела операторов: Program имя;

раздел описаний begin

раздел операторов end.

Заголовок программы начинается со служебного слова Program, после которого указывается имя программы. Раздел описаний предназначен для объявления в программе всех данных, которые встречаются, а также их характеристик.

Существует определенный порядок в разделе описаний:

- раздел меток Label;
- раздел констант Const;
- раздел типов Type;
- раздел переменных Var;
- раздел процедур и функций (Function и Procedure).

Раздел операторов заключается в операторные скобки вида: Begin (начать) и End (закончить), при этом после End ставится точка. Точка – это признак окончания программы. В разделе операторов записывается последовательность выполняемых операторов. Операторы отделяются друг от друга символом ";"

Оператор присваивания – это основной оператор любого языка программирования. Он предназначен для замены текущего значения переменной новым значением.

Форма оператора присваивания имеет вид:

Переменная := выражение;

При выполнении этого оператора значение выражения вычисляется и присваивается переменной.

Например, $a := b + c$;

$w := \sin(\sqrt{t}) / (s + \ln(v))$; $s := \text{„строка“}$;

Имя переменной и результат выражения должны принадлежать к одному типу. Для ввода данных используются операторы:

read (элемент 1, элемент 2, ...);

readln (элемент 1, элемент 2, ...);

При выполнении оператора **read** (элемент1, элемент2, ...) происходят следующие действия: программа приостанавливает свою работу и ждет, пока на клавиатуре будут набраны данные и нажата клавиша Enter. После нажатия клавиши Enter, введенные значения присваиваются переменным, имена которых указаны в операторе **read**. Числовые значения должны быть набраны в одной строке и разделены пробелами.

Оператор **readln** (элемент 1, элемент 2, ...) осуществляет ввод данных, а затем обеспечивает переход к началу новой строки.

Для вывода данных используются операторы:

write (элемент 1, элемент 2, ...);

writeln (элемент 1, элемент 2, ...);

где элемент – это переменная или строка символов, заключенная в апострофы.

Например, оператор **write** („Значение В= „, В);

выводит на экран дисплея текст

Значение В=

а затем числовое значение переменной В.

Для вывода целых и действительных чисел можно указывать форматы в операторе **WRITE**. Формат указывается через двоеточие после переменной. Для действительных чисел формат состоит из двух величин. Первая величина указывает на общее количество позиций для вывода числового значения переменной: знак числа, количество цифр в целой части, точку и количество цифр в дробной части, второе – количество позиций для вывода дробной части. Например, оператор **WRITE (Y:5:2)** осуществит вывод значения Y на экран и при этом для отображения значения Y будет отведено пять позиций, из них две – на дробную часть.

Для вывода целых чисел количество позиций для дробной части не указывается. Например, если необходимо вывести значение целого

числа $N=125$, то оператор вывода будет иметь вид: **WRITE ('N=', N:3)**, т.е. для вывода числового значения N отведено 3 позиции.

Допускается использование оператора вывода **writeln** (элемент1, элемент2 ...), который сначала выводит значения переменных, а затем осуществляет переход на новую строку.

Оператор writeln обеспечивает пропуск одной строки и переход к началу новой строки. Текст программы решения задачи, схема которой приведена в примере имеет вид:

```
Program rabota;
Var a,b,c,p,t,ha,hb,hc:real;Begin

Writeln(„vvod a,b,c“);
Readln(a,b,c);
p:=(a+b+c)/2;
t:=2*sqrt(p*(p-a)*(p-b)*(p-c));ha:=t/a; hb:=t/b;hc:=t/c;
writeln („ha=“,ha:6:2,„ hb=“,hb:6:2, „ hc=“, hc:6:2);end.
```

Исходными данными для решения задачи являются значения длин сторон треугольника: a, b, c. Для ввода этих значений используется оператор Readln. В программе используется переменная p для вычисления полупериметра и вспомогательная переменная t для исключения повторений.

Вычисленные значения высот ha, hb, hc выводятся с соответствующими именами переменных. Когда выводятся элементы списка вывода, пробелы между ними не устанавливаются автоматически, поэтому необходимо непосредственно их указать в списке вывода. Для улучшения внешнего вида данных, которые выводятся на экран дисплея, используется форматированный вывод.

Практические задания

Задание 1.

Изучить лекционный материал и методические рекомендации для выполнения задания.

Разработать алгоритм для расчета заданных переменных.

Составить программу на алгоритмическом языке Pascal.

Выполнить отладку и тестирование программы на компьютере.

Подготовить ответы на контрольные вопросы.

Задание 2. Составить блок-схему и программу вычисления значений функции при заданных значениях аргумента.

$P = \frac{x \sin x}{(2x - y)^2} - \frac{\sqrt{ 2x - y }}{3y \cos y}$	$P = \frac{\sqrt{a^2 + b^2 + c^2}}{2a + b \sin c} - \frac{\ln(a + 2bc)}{a^2 + b^2 + c^2}$
$3) \quad Z = \frac{x^3 + y^2}{y \sin x} - \frac{\ln(y + 5e^{y+x})}{\sqrt[4]{x^3 + y^2}}$	$P = \frac{e^{2a} + 2,5 \sin 3b}{\sqrt{a^2 + b^2}} - 2 \operatorname{tg}(a^2 + b^2)$

$5) \quad Y = \sqrt{a^2 + x^2} + \frac{\operatorname{tg}(a+x) - \ln(a+x)}{\sin x + \cos^2 a}$	$6) \quad Z = \frac{\sin^2(x+y)}{(x+y)^2} - \frac{\sqrt{ 3x-4y }}{\cos x + e^{5x}}$
$Y = \cos x - \frac{\ln x + \sin^2(5x+2)}{5,3x + \frac{7}{\sqrt{2x^2+20}}}$	$P = \frac{\ln a + \ln b}{x + \frac{ax+b}{\sqrt{x^2+2b^4}}} - \operatorname{tg}(ax+b)$
$Y = \frac{\sqrt{ 5x-45 }}{\operatorname{tg} x^2 + \operatorname{tg}^2 x} + \frac{\sqrt{2x^2+6}}{\ln 5x-45 }$	$Y = \cos^3 a + \frac{2ab - \sqrt{a^2+b^2}}{e^{2ab} + \ln a}$
$Z = \frac{\lg(x+y) - x-5y }{\sin 5,3x + \frac{2x}{\sqrt{ x-5y }}} - \frac{\cos^3 y}{x+y}$	$P = \frac{x \sin^y x}{\cos x + e^x} - \sqrt{ \sin^y x + \cos^2 x }$
$13) \quad Z = \frac{y \sin x - \cos x^2 y^3}{\sqrt{ x^y - \cos x + 1 }} + \ln x$	$14) \quad Z = \frac{\cos^2(x+y) - 2x}{\sin^3(x+y) + 3y} - \sqrt{x^4 + y^2}$
$P = 2\operatorname{tg}(a+3b) + \frac{e^{2a} + 2,5 \sin 3b}{\sqrt{ a+3b } - 5}$	$Z = \frac{\sqrt{x^2+y^4}}{\cos^2 x + e^{x+y}} + \frac{\sin^2(x+y)}{2x+y}$
$17) \quad P = \frac{y \sin^2 x}{y^{x-2} + e^{xy}} - \frac{\operatorname{tg} x - 3y}{x-2}$	$Y = \frac{7 \cos a}{\sqrt{ 5a+2b }} - \frac{\sin^2(5a+2b)}{e^{\cos a}}$
$Y = \frac{\sin^2(5x+2)}{5,3x+9} + \frac{7x - e^{2x}}{\sqrt{ 5x+2 }}$	$P = \frac{\sin a + \cos b}{x + \operatorname{tg}(ax+b)} - \frac{a^3 + \sin b}{\sqrt{x^2+7b^2}}$
$Y = \frac{\operatorname{tg} x^2 - \sqrt{2x^2}}{\ln 5x-45 } - \frac{\sqrt{ 5x-45 }}{x + \operatorname{tg}^2 x}$	$Y = \frac{\sqrt{a^2+5b^2}}{e^{2ab} + \ln a} + \cos^3(a^2+5b^2)$
$Z = \frac{\cos y}{\lg x - y-x } + \frac{x+y - \sqrt{ xy }}{5x + \frac{2xy}{\sqrt{ y-x }}}$	$Z = \frac{x \sin x - \cos x}{x^y + \ln x} + \sqrt{ x^y - x \sin x }$
$Z = \frac{\sin^2(x^4+y^2) - 2}{\sin^3(xy) + e^{2y-x}} - \sqrt{x^4+y^2}$	$Y = \frac{2 \sin a}{\sqrt{ a+2b }} - \frac{\cos^2(a+2b)}{e^{\sin a}}$
$P = \frac{\sin(ax+b)}{x + \operatorname{tg}(ax+b)} - \frac{a^3 + \sin b}{\sqrt{x^2+3b^2}}$	$P = \frac{x + \cos x}{(3y-x)^2} - \frac{\sqrt{ 3y-x }}{3y \cos x}$

$P = \frac{\sqrt{a^2 + b^2}}{a + 2bc} - \frac{\ln(a + 2bc)}{a^2 + b^2 + c^2}$	30) $P = \frac{\sin^2 x}{y^{x-2} + e^{xy}} - \frac{x-2}{\cos y^2}$
---	--

Практическая работа №8-14. Составление программ циклической структуры. Обработка одномерных массивов

Цель:

закрепление знаний о программах циклической структуры, составление программы и работа с ней.

Краткие теоретические сведения

Алгоритм циклической структуры - это алгоритм, в котором происходит многократное повторение одного и того же участка программы. Такие повторяемые участки вычислительного процесса называются циклами. Программа циклической структуры содержит один или несколько циклов. Различают детерминированные циклы с заранее известным числом повторений и итерационные циклы, в которых число повторений заранее неизвестно. Изменяющаяся в цикле переменная называется параметром цикла.

Для организации цикла необходимо выполнить следующие действия:

- 1) задать перед циклом начальное значение параметра цикла;
- 2) изменять параметр перед каждым новым повторением цикла;
- 3) проверять условие повторения цикла;
- 4) управлять циклом, т.е. переходить к его началу, если он не закончен, или выходить из него по окончании.

В языке Паскаль существует 3 вида циклов: 1) цикл с параметром или цикл типа for, 2) цикл с предусловием или цикл типа while, 3) цикл с постусловием или цикл типа repeat ... until. В цикле типа for число повторений известно заранее, в циклах типа while и repeat ... until число повторений цикла заранее неизвестно, производится проверка условия повторения цикла: в цикле типа while

- перед циклом, в цикле типа repeat ... until - после его окончания.

В циклах типов for и while повторяющаяся часть (тело цикла) состоит из одного оператора, если требуется выполнить в цикле несколько операторов, они заключаются в операторные скобки begin ... end, образуя составной оператор. В цикле типа repeat ... until тело цикла помещается между зарезервированными словами языка (лексемами) repeat и until, операторные скобки не требуются, в названии цикла его тело условно обозначается тремя точками.

С помощью цикла типа for удобно находить суммы, произведения, искать максимальные и минимальные значения и т.п. При нахождении суммы некоторой переменной, например S присваивается значение 0, затем в цикле к этой переменной

прибавляется соответствующий член заданной последовательности. При нахождении произведения переменной присваивается значение 1, затем в цикле эта переменная умножается на общий член последовательности.

Пример цикла типа for. Вычисление n чисел Фибоначчи: $F_1=1$; $F_2=1$;...;
 $F_n=F_{n-1}+F_{n-2}$,

например $F_3=F_2+F_1=1+1=2$; $F_4=2+1=3$ и т.д. program fib; {Нахождение чисел Фибоначчи}

```
var x, y, z, i, n : integer; begin
writeln ('Введите n'); read (n); x:=1; y:=0;
for i:=1 to n do begin
z:=x ; x:=x+y ; y:=z ;
write (' ', x ) ; end;
end.
```

Пример цикла типа while. Составление таблицы функции $y = a^3/(a^2+x^2)$ для x, принадлежащих отрезку $[-1; 1]$ с шагом 0.1. Так как параметр цикла типа for должен быть целочисленным, удобнее использовать цикл while, в котором значение x можно изменять при каждом шаге на $x = 0.1$

```
program cycl_while;
uses crt; {вызов модуля Crt для управления режимом экрана} var
a, x, y : real ; i : integer; begin
clrscr ; {процедура очистки экрана из модуля Crt} writeln ('Введите a '); readln
(a);
x:= -1.; writeln (' x ', ' y '); while do x<1.05 begin
y:= sqr(a)*a/ (a*a+x*x); writeln (x:6:2, y:8:4); x:=x+0.1
end end .
```

Условие $x<1.05$ соответствует каждому значению x плюс половина шага .

Пример цикла типа repeat ... until. Определить число n, при котором сумма квадратов натурального ряда чисел от 1 до n не превысит величину K, введенную с клавиатуры. Т.е.

$S \leq K$, где $S = 1^2 + 2^2 + \dots + n^2$

```
program sum_sq; {Сумма квадратов натурального ряда } uses crt;
var k, s, n : integer; begin
clrscr ; writeln( 'Введите K' ); readln (k); s:=0; n:=1;
repeat
s :=s+n*n; n := n+1; until s > k;
writeln ('N= ', n : 3, ' s= ', s : 5 ); end.
```

Цикл повторяется до тех пор, пока условие записанное после ключевого слова

until, будет ложным (не выполняется). Как только это условие выполнится, происходит выход из цикла. По-сле окончания цикла производится печать результата (оператор writeln).

Отметим, что цикл с предусловием (типа while) может не выполниться ни разу, цикл с по-стусловием repeat ... until выполнится по крайней мере 1 раз.

Когда число повторений цикла неизвестно заранее, применяются циклы с предусловием или с постусловием. Когда число повторений цикла известно заранее, как правило, применяется цикл типа for. Но любой цикл типа for можно заменить циклом с предусловием или постусло-вием.

Практические задания

Задания

1. Разобрать и проанализировать приведенные выше программы с циклами типа for, while и repeat ... until
2. Составить алгоритм задачи: Вычислить сумму ряда, указанного в варианте задания для лю-бого значения N, введенного с клавиатуры.
3. Составить 3 варианта программ циклической структуры типа for, while и repeat, откомпили-ровать их, ввести исходные данные, сравнить полученные результаты.

Варианты заданий

$$\sum_{i=1}^N \frac{1+i}{2+i^2} \quad \sum_{i=1}^N \frac{6}{4+i^3}$$

. 2. .

.

$$\sum_{i=1}^N \frac{3i^2}{24+i} \quad \sum_{i=1}^N \frac{8+3i^2}{12+i^2}$$

.

$$\sum_{i=1}^N \frac{8}{16+i^2} \quad \sum_{i=1}^N \frac{3+i}{20+i}$$

.

$$\sum_{i=1}^N \frac{12-i^2}{6+i} \quad \sum_{i=1}^N \frac{3-i}{2+8i^2}$$

.

Составление программ обработки одномерных массивов.

Краткие теоретические сведения

При выполнении работ использоваться структурированный тип данных *массив*, который представляет собой фиксированное количество упорядоченных однотипных компонент, снабженных индексами. Он может быть *одномерным* и *многомерным*.

Тип-массив описывается в разделе описания типов следующим образом:

type <имя типа> = **array**[<тип индекса(индексов)>] **of** <тип компонент>;

Размерность массива может быть любой, компоненты массива могут быть любого (в том числе и структурированного) типа, индекс (индексы) может (могут) быть только интервального или перечисляемого типа. При употреблении в качестве типа индекса типа *Integer* или *Word* можно использовать только его диапазон.

При описании типа индекса (индексов) можно использовать константы, которые должны быть определены до определения типа.

Определенный в разделе описания типов тип-массив можно использовать для описания переменных и типизированных констант. Тип-массив можно вводить непосредственно и при определении соответствующих переменных или типизированных констант.

При задании значений константе-массиву компоненты указываются в круглых скобках и разделяются запятыми, причем, если массив многомерный, внешние круглые скобки соответствуют левому индексу, вложенные в них круглые скобки – следующему индексу и т. д. При этом все компоненты массива должны быть заполнены.

Доступ к компонентам массива осуществляется указанием имени массива, за которым в квадратных скобках помещается значение индекса (индексов) компоненты. В многомерных массивах значения индексов перечисляются через запятую.

Для работы с массивом как с единым целым используется идентификатор массива без указания индексов в квадратных скобках. Массив может участвовать только в операциях отношения («=», «<>») и в операторе присваивания.

Массивы, участвующие в этих действиях, должны быть идентичны по структуре, то есть иметь одинаковые базовые типы и типы индексов.

Задание значений переменной типа массив может осуществляться с клавиатуры, путем генерирования случайным образом либо иным способом. Когда значения элементов задаются с клавиатуры или генерируются случайным образом, необходима организация цикла, в котором последовательно происходит обращение к каждому элементу массива. Вывод значений элементов массива на экран или в файл также происходит в цикле.

При написании программ необходимо следить за тем, чтобы значения индексов не превышали границ, указанных при объявлении массива, так как выход индекса за границы массива приводит к сбою в работе программы. Контроль значений индексов массивов можно организовать при помощи директивы компилятора `{R+}`, которая приводит к проверке всех индексных выражений на соответствие их значений диапазону индекса.

Пример выполнения лабораторной работы

Для примера рассмотрим следующую задачу. Дан одномерный массив целых чисел. Удалить все отрицательные элементы, расположенные справа от максимального.

Прежде всего, определим алгоритм решения данной задачи. Вначале надо найти максимальный элемент среди элементов массива и запомнить его индекс. Затем, начиная со следующего за максимальным элементом, проверять элементы массива на знак, и если число оказывается отрицательным, то удаляем этот элемент из массива путем «сдвига» элементов массива влево и уменьшаем длину массива. При этом следует заметить, что в памяти переменная типа массив будет занимать столько же места, как и до удаления элементов.

В графическом виде этот алгоритм будет блок-схему, представленную на рис.

5.1. Теперь пишем код программы.

Поскольку в программе требуется использование генератора случайных чисел, то необходимо

подключить стандартный модуль CRT, в котором описана функция генерации случайных чисел.

Uses crt;

В разделе описания типов определим тип-массив:

type arr=array[1..15] of integer;

В разделе описания переменных описываем необходимые переменные: var
a:arr; {массив}

i,j,n: byte; {индексы, длина массива}

i_max: byte; {индекс максимального элемента массива} max:integer;

{максимальный элемент}

Вторая часть алгоритма – поиск максимального элемента. Предположим, что первый элемент – максимальный, тогда запомним значение первого элемента в переменной max, а в переменную i_max, где будет запоминаться положение максимального элемента, запишем 1. Далее организуется цикл, в котором ищется элемент, больший чем max.

max:=a[1]; i_max:=1; for i:=2 to n do

if a[i]>max then begin

max:=a[i]; i_max:=i; end;

Далее, начиная с элемента, следующим за максимальным, организуется цикл проверки эле-

ментов массива на знак (важно заметить, что в данном случае нужно использовать цикл while, т. к. длина массива будет изменяться, и, соответственно, число итераций цикла также будет меняться). Если число оказывается отрицательным, то удаляем этот элемент из массива, для чего организуется еще один цикл, в котором на место удаляемого записывается стоящий следом элемент.

Внутренний цикл сдвига можно проиллюстрировать на примере удаления

седьмого элемента из массива, содержащего 10 элементов (рис. 5.2).

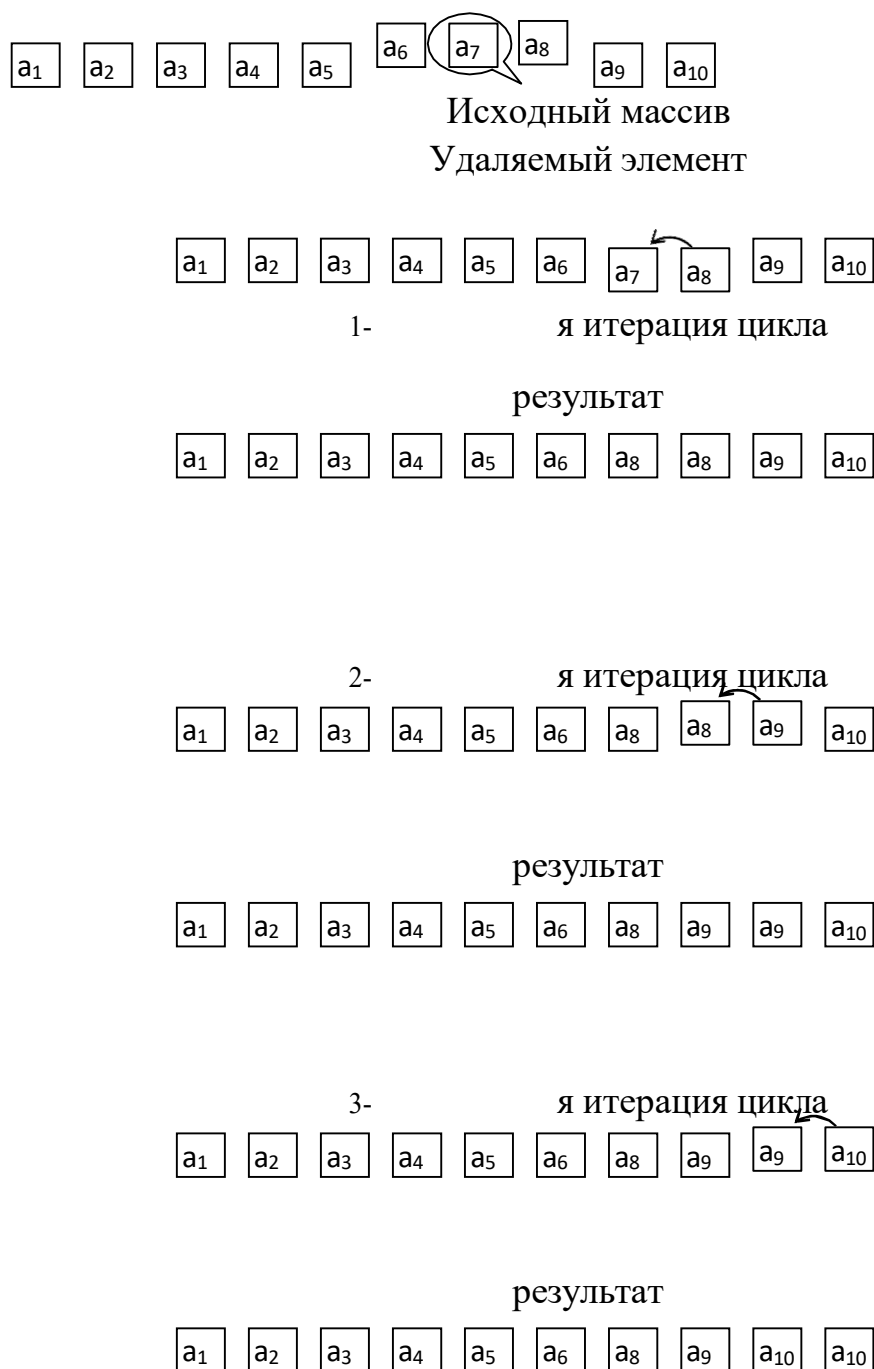


Рис. 5.2. Схема удаления элемента из массива

В программе удаление элементов, находящихся справа от максимального, будет выглядеть следующим образом:

```

i:=i_max; while i<=n do begin if a[i]<0 then begin
for j:=i to n-1 do a[j]:=a[j+1]; n:=n-1;
end else
i:=i+1;
end;

```

Еще одна тонкость данного фрагмента состоит в том, что переход к следующему элементу ($i:=i+1$) для проверки осуществляется только тогда, когда элемент не удовлетворяет условию. Если же i -й элемент окажется отрицательным, то после удаления на его месте окажется $i+1$ -й элемент, который и нужно проверять на следующей итерации цикла.

Остается вывести результирующий массив на экран. `writeln('Результат обработки:');` for $i:=1$ to n do

```

write(a[i]:4); readln;
{$R-}                                {Выключаем контроль значений индексов} end.

```

Ниже представлен полный текст программы. `program lab5_var100; Uses crt;`

```

type arr=array[1..15] of integer; var a:arr;           {массив} i,j,n: byte; {индексы,
длина массива}

```

```

i_max: byte; {индекс максимального элемента массива} max:integer;
{максимальный элемент}

```

```
begin writeln;
```

```
writeln('                                Автор – Иванов И.П., студент гр. ИСЭд-11');
```

```
writeln(' Вариант № 100');
```

```
writeln(' Дан одномерный массив целых чисел.');
```

```
writeln(' Удалить все отрицательные элементы, расположенные справа от
максимального
```

Варианты заданий

Скорректировать массив $A=(a_1, a_2, \dots, a_n)$, переписав в начало массива

группу, содержащую наибольшее число подряд идущих положительных элементов. Элементы массива вводить с клавиатуры.

1. В массиве $A=(a_1, a_2, \dots, a_n)$ все элементы, равные нулю, поставить сразу после максимального элемента данного массива. Элементы массива вводить с клавиатуры.

2. В массиве $A=(a_1, a_2, \dots, a_n)$ все отрицательные элементы отправить в «хвост» массива.

3. В массиве $A=(a_1, a_2, \dots, a_n)$ удалить последнюю группу положительных элементов. Группой называется подряд идущие элементы одного знака, число которых больше или равно 2.

4. В массиве $A=(a_1, a_2, \dots, a_n)$ все положительные элементы, стоящие перед минимальным положительным элементом, переслать в «хвост» массива.
5. В массиве $A=(a_1, a_2, \dots, a_n)$ удалить все подряд идущие отрицательные элементы, идущие вслед за минимальным элементом массива.
6. В массиве $A=(a_1, a_2, \dots, a_n)$ удалить все отрицательные элементы, стоящие перед минимальным элементом массива.
7. В массиве $A=(a_1, a_2, \dots, a_n)$ удалить все элементы, меньшие, чем элемент массива, расположенный слева от максимального.
8. В массиве $A=(a_1, a_2, \dots, a_n)$ вставить новый элемент со значением P вслед за наибольшим из отрицательных элементов этого массива.
9. В массиве $A=(a_1, a_2, \dots, a_n)$ удалить все элементы, стоящие между минимальным положительным и максимальным отрицательным элементами.
10. В массиве $A=(a_1, a_2, \dots, a_n)$ удалить все положительные элементы, имеющие четный порядковый номер, идущие после минимального элемента массива.
11. В массиве $A=(a_1, a_2, \dots, a_n)$ все положительные элементы, начиная со второго положительного, отправить в «хвост» массива.
12. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ группу, содержащую наибольшее число равных элементов, заменить на максимальный элемент этого массива. После корректировки массив может содержать меньше элементов, чем прежде. Элементы массива вводить с клавиатуры.
13. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ группу элементов, содержащую наибольшее число подряд идущих отрицательных элементов, переписать в «хвост» массива. Элементы массива вводить с клавиатуры.
14. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ все отрицательные элементы, имеющие нечетный порядковый номер, отправить в «хвост» массива, т. е. поместить на место последних элементов.
15. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ все группы элементов, содержащие более 3-х подряд идущих отрицательных элементов, заменить на максимальный элемент. Элементы массива вводить с клавиатуры.
16. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ все положительные элементы, имеющие четный порядковый номер, переписать в начало массива.
17. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ группу, содержащую наибольшее число равных элементов, заменить на максимальный элемент этого массива. Элементы массива вводить с клавиатуры.
18. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ удалить все отрицательные элементы, расположенные между положительными.

19. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ исключить из массива группу с наибольшим числом подряд идущих положительных элементов. Элементы массива вводить с клавиатуры.

20. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ отрицательные элементы, имеющие четный порядковый номер, переписать в начало массива.

21. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ удалить все равные элементы, оставив только один из данных групп равных. Элементы массива вводить с клавиатуры.

22. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ группу из наибольшего числа подряд идущих нулей заменить на максимальный элемент массива. Элементы массива вводить с клавиатуры.

23. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ группу, содержащую наибольшее число подряд идущих положительных элементов, переписать в «хвост» массива.

24. В одномерном массиве $A=(a_1, a_2, \dots, a_n)$ все положительные элементы, расположенные между отрицательными, поставить после минимального элемента массива.

Составление программ обработки двумерных массивов

Цель работы: овладеть основными приемами работы с одномерными и двумерными массивами, уметь различать в двумерном массиве обработку строк и столбцов, а также отличать нахождение первых и последних элементов последовательности, обладающих некоторым свойством.

Общие сведения

Скалярный тип – простой тип данных. Скалярное данное неделимо. *Массив* – это структурированный тип данных. Массив состоит из нескольких элементов. Ко всему массиву можно обращаться по его имени. Можно обращаться к его элементу, но для этого надо задать индекс (индексы). Массивы бывают одномерные и многомерные. Для объявления массива необходимо задать типы его индексов и компонент:

ARRAY [Тип индексов] OF <Тип компонент>;

Тип компонент массива – это просто тип данных, ассоциированный с каждой компонентой массива. Тип компонент может быть любым REAL, INTEGER, CHAR, BOOLEAN, перечислимый, интервальным. В качестве компоненты массива может быть взят и тип массив.

Тип индекса должен быть одним из упорядоченных типов, т.е. любым

скалярным типом, кроме REAL: INTEGER, CHAR, интервальный, перечислимый. Тип индекса определяет границы изменения индекса. Если сделана попытка использовать несуществующую компоненту, то возникает ошибка (ошибка неверного индекса).

Одномерные массивы

Одномерный массив можно задать (объявить) двумя способами:

1. С помощью служебного слова TYPE описывается тип массива, а затем с помощью VAR вводится переменная этого типа.

Общая форма записи

TYPE <тип массива> = ARRAY [тип индекса] OF <тип компонент>; VAR
<переменная>: <тип массива>;

2. С помощью слова VAR сразу описывается переменная типа массив.

Общая форма записи

VAR <переменная>: ARRAY [тип индекса] OF <тип компонент>;

Например, объявление массива из 100 элементов типа REAL можно осуществить двумя способами:

1. type R100 = array [1..100] of real; var A: R100;

2. var A: array [1..100] of real.

Здесь задан массив с именем A, и его элементы имеют имена: A[1],...,A[100].

Чаще всего для типа индекса используют интервальный тип на основе типов INTEGER и CHAR. Однако можно в качестве индексов брать перечислимый тип.

Пример 1. Подсчет числа вхождений букв в текст определенной длины.
program COUNTER;

```
var COUNT: array ['a'..'z'] of integer; CH: char; N: integer;
begin
for CH := 'a' to 'z' do COUNT [CH] := 0; N := 0;
repeat
read (CH); N := N + 1;
if (CH >= 'a') and (CH <= 'z') then COUNT [CH] := COUNT [CH] + 1;
until CH = '.';
for CH := 'a' to 'z' do
writeln (CH, COUNT [CH]:5);
end.
```

Пояснение. В этом примере тип индекса есть интервальный тип на базе типа CHAR, а тип компонент есть целое число. Таким образом, элементы массива – числа, а их индексы – буквы, т.е. число элементов массива равно 26 (по числу букв

латинского алфавита).

Рассмотрим теперь случай, когда тип индекса задан перечислимым типом, а компоненты массива представлены компонентами интервального типа на базе типа INTEGER.

Пр и м е р 2. Присваивание переменной с именем месяца числа дней этого месяца. program NUMBRDAY;
type MONAT = (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OKT,
NOV, DEC);
var DAY : array [MONAT] of 28..31; T : MONAT;
begin
for T := JAN to DEC do case T of
JAN, MAR, MAY, JUL, AUG, OKT, DEC: DAY [T] := 31; APR, JUN, SEP, NOV:
DAY [T] := 30;
FEB : DAY [T] := 28;
end; end.

Многомерные массивы

Для определения позиции элемента в двумерном массиве необходимы два индекса. Любой двумерный массив есть матрица, а матрица есть таблица. Поэтому удобно описывать двумерные массивы путем указания границ изменения индексов (номеров) строк и столбцов.

Например, таблица символов $M \times N$, где M – число строк и N – число столбцов, может быть описана:

```
var TAB : array [1..M, 1..N] of char;
```

Общая форма записи

```
VAR <имя> : ARRAY [тип индекса строки, тип индекса столбца] OF <тип  
компонент>;
```

Однако двумерный массив можно интерпретировать как вектор-столбец, каждый элемент которого, в свою очередь, является одномерным массивом (вектор-строка). Этот подход к определению двумерного массива влечет его описание с помощью двух строк, где первая содержит описание строки, а вторая – описание столбца:

```
type LINE = array [1..N] of char; STOLB = array [1..M] of LINE;  
var TAB : STOLB.
```

Здесь TAB [I] – переменная типа LINE, а TAB [I][J] – переменная типа CHAR.

Общая форма записи

```
TYPE <тип строки> = ARRAY [тип индекса] OF <тип компонент>;  
<тип столбца> = ARRAY [тип индекса] OF <тип строки>; VAR <переменная  
массива> : <тип столбца>;
```

Эти два вида определения массивов задают и два способа обращения к элементам массива: TAB [I, J] – в первом случае и TAB [I][J] – во втором. Вполне очевидно, что сказанное выше для двумерного массива распространяется и на массивы большей размерности. Например, описание VAR CUBE : ARRAY [1..M, 1..N, 1..K] OF INTEGER определяет задание трехмерного массива целых чисел.

Примеры работы с массивами

Обработка массивов включает в себя, как правило, следующие компоненты: ввод массива (с клавиатуры или с помощью датчика случайных чисел), вывод полученного массива на экран и собственно его обработка. Все эти компоненты рекомендуется оформлять в виде отдельных процедур. При этом надо учитывать следующий фактор: если процедуре (или функции) будет передаваться массив, то надо объявить в ней этот массив как параметр с атрибутом VAR даже в том случае, если значение массива внутри процедуры не изменяется. Это нужно для того, чтобы не тратить времени и памяти на размещение внутри процедуры копии массива. Заметим, что параметр обязательно должен относиться к типу, имеющему имя.

Пр и м е р. Сумма элементов таблицы над верхней диагональю. program SUMMA;

```
const M = 10; {число строк таблицы} N = 10; {число столбцов таблицы} type  
LINE = array [1..n] of integer; TAB = array [1..m] of LINE;  
var s, i, j:integer; MAS: TAB;  
procedure VVODMASSIV (var MAS: TAB); begin  
for i := 1 to M do for j := 1 to N do  
readln (MAS [i][j]); end;  
procedure VIVODMASSIV (var MAS:TAB); begin  
for i := 1 to M do begin  
for j := 1 to N do  
write (MAS [i][j]:4, ' '); writeln; end;
```

Пример 1. Составить программу, позволяющую в одномерном массиве, состоящем из N вещественных элементов, вычислить сумму положительных элементов.

Решение. При написании процедур ввода и вывода следует обратить внимание, что элементы – вещественные числа, поэтому необходимо позаботиться о верной обработке дробной части. Вычисление суммы оформим в виде функции с одним аргументом - массивом. Локальными переменными функции будут индексная переменная i и дополнительная переменная s для хранения текущей суммы элементов. В начале тела функции обязательно обнуление s . Каждый элемент массива сравним с нулем, и, если значение положительно, добавим его к искомой сумме s . В конце функции запишем значение переменной s в результирующую переменную.

```
program massiv_1; const N=10;
type mas=array [1..N] of Real; procedure Vvodmas(var A:mas); var i:Integer;
begin
for i:=1 to N do A[i]:=-50+Random(101)+random; end;
procedure Vivodmas(A:mas); var i:Integer;
begin
for i:=1 to N do Write(A[i]:8:2);
Writeln end;
function Summa(A:mas):real; var i: Integer; s:real;
begin s:=0;
for i:=1 to N do if A[i]>0 then s:=s+A[i]; Summa:=s;
end;
var A: mas; begin
Randomize; Vvodmas(A); Writeln('Исходный массив:'); Vivodmas(A);
Writeln('Ответ: ', Summa(A):0:2);
Readln end.
```

Пример 2. В двумерном массиве, состоящем из целочисленных элементов, в каждом столбце поменять местами наибольший по модулю и последний не принадлежащий интер-валу (a, b) элементы массива.

Решение. Преобразования необходимо провести в каждом столбце массива, поэтому па-раметр внешнего цикла в процедуре обработки - номер столбца j , а вложенного – номер строки i .

Для перестановки двух элементов в столбце массива необходимо найти номера их строк $n1$ и $n2$, а затем поменять местами значения элементов с использованием промежуточной переменной p .

Чтобы найти наибольший по модулю элемент столбца, введем дополнительную переменную *max*, которая будет хранить максимальное по модулю значение в текущем столбце массива на данный момент. (Можно решить задачу без использования переменной *max*. Подумайте, как это сделать.)

Программа должна корректно работать с любыми входными данными, а значит и в тех случаях, когда некоторые или даже все столбцы массива содержат только элементы из интервала (a, b) , и обмен значений в некоторых столбцах или во всем массиве не нужен.

```
const n=10; m=7;
type mas=array [1..n,1..m] of Integer; procedure Vvodmas(var D:mas);
var i,j:Integer; begin
for i:=1 to n do for j:=1 to m do
D[i,j]:=-50+Random(101);
end;
procedure Vivodmas(D:mas); var i,j:Integer;
begin
for i:=1 to n do begin
for j:=1 to m do Write(D[i,j]:4); Writeln;
end; end;
procedure Obmen(a,b: real; var D:mas); var i,j,p,n1,n2,max: Integer;
begin
for j:=1 to m do begin
n1:=1; max:=abs(D[1,j]); {считаем первый элемент столбца наибольшим по
модулю} for i:=2 to n do
if abs(D[i,j])>max then {обнаружен больший элемент} begin n1:=i;
max:=abs(D[i,j]) end;
i:=n; {перебираем элементы столбца, начиная с последнего} while (i>=1)and
(D[i,j]>a)and(D[i,j]<b) do i:=i-1;
n2:=i;
if n2<>0 then {если элемент, не принадлежащий интервалу (a,b), был найден}
begin
p:=D[n1,j]; D[n1,j]:=D[n2,j]; D[n2,j]:=p; {обмен значений} end;
end; end;

var D: mas; a,b:Real; begin
Randomize; Vvodmas(D); Writeln('Исходный массив:'); Vivodmas(D);
Write('Введите через пробел концы интервала (a,b): '); Readln(a,b);
```

```
Obmen(a,b,D);
  Writeln('Ответ:'); Vivodmas(D);
  Readln end.
```

Варианты заданий

Задание 1. Составить программу, позволяющую в одномерном массиве, состоящем из N

вещественных элементов, вычислить:

- a) сумму модулей отрицательных элементов массива;
- b) количество элементов массива, не принадлежащих интервалу (a, b) ;
- c) наименьший из элементов массива, принадлежащих отрезку $[a, b]$;
- d) количество элементов массива, равных первому элементу;

Обработка двумерных массивов

2.1. Пример обработки многомерного массива

Дана матрица $A(3,4)$, и вектор $B(4)$, состоящие из целых чисел. Умножить матрицу A на вектор B .

```
program pr4-2 ; const m=3; n=4; var
a : array [ 1 .. m, 1 .. n ] of integer; (* описание матрицы *)
b : array [ 1 .. n ] of integer; (* описание вектора *)
c : array [ 1 .. m ] of integer; (* описание C *) i, j: integer;
begin
  for i:=1 to m do (* ввод матрицы *) begin
    writeln ('введите элементы ', i, '-той строки'); for j:=1 to n do read (a [i, j] );
    writeln;
  end;
  writeln ('введите элементы вектора'); for j:=1 to n do (* ввод вектора *) read (b[
j]); writeln;
  for i:=1 to m do

    begin
      c [ i ]:=0; for j:=1 to n do c[i] := c[ i ]+ a[i , j]* b[j]; end;
      for i:=1 to m do (*форматный вывод матрицы *)

        begin
          for j:=1 to n do write (a [i, j]: 4 ); writeln; end;
          for j:=1 to n do write (b [ j ]:4); (* вывод массива B *) writeln ;
          for i:=1 to m do write (c [ i ] :4); (* вывод массива C *) readln;
          end.
```

В программе элементы матрицы вводятся по строкам по одному с

подтверждением клавишей Enter. А выводятся в общепринятом виде: каждая строка матрицы с новой строки экрана (цикл i по строкам внешний, а цикл j – внутренний).

Практические задания

Задание 1.

1. Набрать и откомпилировать приведенные выше программы, исправить выявленные ошибки. Ввести элементы массива, убедиться в правильности выполнения программ.

2. Составить и выполнить программы с применением массивов согласно вариантам заданий.

Задание 2.

Варианты задания 1. Ввести массив A из 10 элементов

1. Найти наибольший элемент и переставить его с первым элементом. Преобразованный массив вывести.

2. Найти наименьший элемент и переставить его с последним элементом. Преобразованный массив вывести.

3. Найти произведение положительных элементов и вывести его на экран.

4. Найти произведение отрицательных элементов и вывести его на экран.

5. Найти сумму положительных элементов и вывести ее на экран.

6. Найти сумму отрицательных элементов и вывести ее на экран.

7. Найти сумму элементов, больших 3 и меньших 8 и вывести ее на экран.

8. Найти сумму элементов, меньших по модулю 5 и вывести ее на экран.

Задание 3.

Варианты задания 2

1. Даны матрица A размером $m \times n$ и вектор B размером m . Записать на главную диагональ элементы вектора, а в вектор – элементы главной диагонали.

2. Выбрать максимальный элемент матрицы C (размер $m \times n$), элементы четных строк разделить на максимальный элемент, а к элементам нечетных прибавить максимальный элемент.

3. Найти минимальный элемент матрицы C (размер $m \times n$), и поменять его местами с первым элементом.

4. Дана матрица E размером $m \times n$. Вычислить суммы элементов каждого столбца. Определить наибольшее значение этих сумм и номер соответствующего столбца.

5. В матрице K размером $m \times n$ найти в каждом столбце

произведение отрицательных элементов и количество нулевых элементов в матрице.

6. Даны две матрицы A и B одинаковой размерности $m \times n$. Получить матрицу $C = \max(a_{ij}, b_{ij})$, и матрицу $D = \min(a_{ij}, b_{ij})$.

7. Дана матрица P размером $m \times n$. Найти сумму минимальных элементов каждого столбца матрицы.

8. Даны две матрицы: A размером $m \times k$ и B размером $k \times n$. Получить матрицу $C = A * B$.

Практическая работа №15-24.

Краткие теоретические сведения

Наряду с числовой информацией в Паскале используется алфавитно-цифровая или символьная информация, которая включает в себя заглавные, строчные буквы, цифры от 0 до 9 и вспомогательные символы. Для описания символьных переменных используется тип данных CHAR или STRING.

2.1. Тип данных CHAR

Каждая переменная символьного типа может принимать значение только одного символа. Все символы упорядочены в соответствии с принятым в ЭВМ коде (например ASCII). При этом порядковый номер символов называется кодом (например, код латинского символа 'A' равен 65; символа 'Z' равен 90).

Для символьных данных не определены никакие арифметические операции, но они могут сравниваться по своим кодам, участвовать в чтении, печати, операторах присваивания. Существуют две стандартные функции преобразования :

1) ORD (C) принимает значение кода символа C;

2) значение функции CHR(I) является символ с кодом

Например: $\text{ORD}('A') = 65$ $\text{CHR}(\text{ORD}(C)) = C$; $\text{CHR}(65) = A$;

Строка - это последовательность символов. Строку можно представить как массив, элементы которого имеют тип CHAR. Например:

BUK: array[1..17] of char;

Массив BUK-массив символов, который содержит 17 символов. Если символов меньше, то строка дополняется пробелами справа. В противном случае возникает ошибка несоответствия типов. Так как массивы символов являются обычными массивами с элементами типа CHAR, они обладают всеми свойствами массивов.

Пример: Из набора 10 любых символов напечатать только заглавные английские буквы и их коды.

```
program lr2;
```

```
type sl = array [1..10] of char;
```

```
var s: sl; {описание массива символов} i: integer;
```

```
begin
```

```
writeln ('введите 10 символов');
```

```
for i:=1 to 10 do readln (s[i]);
```

```
{ввод массива} for i:=1 to 10 do
```

```
if (s[i]>='A') and (s[i]<='Z') then  
  writeln ('Символ :', s[i], ' его код =', ord (s[i])); readln;  
end.
```

2.2. Тип данных STRING

В Турбо Паскале предусмотрен тип данных STRING. Переменная типа STRING может принимать значения переменной длины. Максимально возможная длина переменной 255 символов.

Например:

```
str: STRING[200];  
ow: STRING[10];
```

В скобках указывается максимальная длина для данной переменной. Для ввода значений типа STRING необходимо использовать READLN, а не READ. За один раз может быть введена только одна строка. Две строки можно сравнивать, используя операции отношения (сначала сравниваются самые левые символы, если они равны, то сравниваются следующие). Для работы

с переменными типа STRING используют следующие стандартные процедуры и функции:

1) Функция LENGTH

C:=LENGTH(str); Переменной C будет присвоено целое значение, показывающее количество символов в строковой переменной str .

2) Функция CONCAT - сцепление строк в порядке их перечисления.

str:=CONCAT(st1,st2,...,stN); str-переменная типа STRING, состоящая из строк st1,...,stN.

3) Функция POS

P:=POS (st1, st2); P-целое число, показывающее номер позиции, с которой начинается строка st1 в строке st2.

4) Функция COPY

S1:=COPY(str, I, J); S1-символьная подстрока, выделенная из строки str с позиции I, дли-ной J символов.

5) Процедура DELETE(Str, I, J);

Из строки str удаляется J символов, начиная с I позиции.

6) Процедура INSERT(Str1, Str2, I);

Строка Str1 вставляется с I позиции в строку Str2.

7) Процедура STR (V, S1);

Числовое значение переменной V преобразуется в строку символов и записывается в строку S1.

7) Процедура VAL (S1, V, C);

Строковое выражение S1 преобразуется в величину целочисленного или вещественного типа и записывается в переменной V . Если при этом ошибок не обнаруживается, то C будет равно 0 . В противном случае значение C будет равно номеру позиции первого ошибочного сим-вола и V будет неопределено. Строка S1 не должна содержать незначащих пробелов, переменная V может быть целой или

вещественной, а переменная С - только целой .

Пример: Подсчитать количество слов во введенной с клавиатуры строке.
program lr2;

```
var
s: string[30];
kol, i, n: integer;
begin
writeln ('введите строку'); readln (s); kol:=0;      {счетчик количества слов}
n:= length(s);      {определяем длину введенного текста} s:= concat(' ',s);
{добавляем пробел к первому слову}
for i:=1 to n do
if (copy (s,i,1)=' ') and (copy (s,i+1,1)<>' ')
then kol := kol+1; {подсчет количества слов} writeln (s, ' количество слов= ',
kol);
```

Практические задания

Задание

1. Набрать и откомпилировать приведенные выше программы, исправить выявленные ошибки. Ввести разные исходные данные, убедиться в правильности выполнения программ.

2. Составить программы с использованием символьных данных согласно вариантам заданий, откомпилировать их, проверить полученные результаты.

Задание 2.

Варианты заданий. Текст вводится с клавиатуры в символьную переменную. Исходный

текст и результаты распечатать.

1. Вывести на печать список слов, имеющих приставку (несколько букв), задаваемую с терминала.
2. Раздвинуть заданный текст, вставив введенную с клавиатуры последовательность символов после 1-го символа каждого слова.
3. В заданном тексте слова разделены запятыми. Напечатать список слов, начинающихся с символа, введенного с клавиатуры.
4. Определить наличие слов в заданном тексте, содержащих сочетание символов, задаваемое с экрана.
5. Из заданной последовательности слов удалить слова, содержащие числа.
6. Каждое слово текста преобразовать таким образом, чтобы оно читалось слева направо.
7. Подсчитать количество слов, разделенных запятыми,

содержащих k гласных букв (k-задается с экрана).

8. Из введенного текста сначала распечатать слова, заканчивающиеся на согласную букву, а потом на гласную букву.

Работа с данными типа множество

Цель:

познакомить с понятием "множество" в языке программирования Pascal; выработать навыки работы со структурой данных множество.

Практические задания

Перед выполнением работы необходимо ознакомиться с правилами описания и использования переменных типа множество, типизированных констант типа множество, переменных, заданных перечислением, изучить допустимые операции над переменными этих типов.

Понятие множества в языке ПАСКАЛЬ основывается на математическом представлении о множествах: это ограниченная совокупность различных элементов. Для построения конкретного множественного типа используется перечисляемый или интервальный тип данных. Тип элементов, составляющих множество, называется базовым типом.

Множественный тип описывается с помощью служебных слов Set of, например: type M= Set of B;

Здесь M - множественный тип, B - базовый тип. Пример описания переменной множественного типа:

```
type  
M= Set of 'A'..'D';  
var  
MS : M;
```

Принадлежность переменных к множественному типу может быть определена прямо в разделе описания переменных:

```
var  
C : Set of 0..7;
```

Константы множественного типа записываются в виде заключенной в квадратные скобки последовательности элементов или интервалов базового типа, разделенных запятыми, например:

['A', 'C'] [0, 2, 7] [3, 7, 11..14].

Константа вида [] означает пустое подмножество.

Множество включает в себя набор элементов базового типа, все подмножества данного множества, а также пустое подмножество. Если базовый тип, на котором строится множество, имеет K элементов, то число подмножеств, входящих в это множество, равно 2^K . Пусть имеется переменная P интервального типа:

```
var
```

P : 1..3;

Эта переменная может принимать три различных значения - либо 1, либо 2, либо 3. Переменная T множественного типа

var

T : Set of 1..3;

может принимать восемь различных значений: [] [1,2]

[1] [1,3]

[2] [2,3]

[3] [1,2,3]

Порядок перечисления элементов базового типа в константах безразличен.

Значение переменной множественного типа может быть задано конструкцией вида [T], где T - переменная базового типа.

К переменным и константам множественного типа применимы операции присваивания (:=), объединения(+), пересечения(*) и вычитания(-):

['A','B'] + ['A','D'] даст ['A','B','D']

['A'] * ['A','B','C'] даст ['A']

['A','B','C'] - ['A','B'] даст ['C'].

Результат выполнения этих операций есть величина множественного типа.

К множественным величинам применимы операции: тождественность (=), нетождественность ($\neq$), содержится в ($\subseteq$), содержит ($\supseteq$). Результат выполнения этих операций имеет логический тип, например:

['A','B'] = ['A','C'] даст FALSE

['A','B'] $\neq$ ['A','C'] даст TRUE

['B'] $\subseteq$ ['B','C'] даст TRUE

['C','D'] $\supseteq$ ['A'] даст FALSE.

Кроме этих операций для работы с величинами множественного типа в языке ПАСКАЛЬ используется операция in проверяющая принадлежность элемента базового типа, стоящего слева от знака операции, множеству, стоящему справа от знака операции. Результат выполнения этой операции - булевский. Операция проверки принадлежности элемента множеству часто используется вместо операций отношения, например:

При использовании в программах данных множественного типа выполнение операций происходит над битовыми строками данных. Каждому значению множественного типа в памяти ЭВМ соответствует один двоичный разряд. Например, множество

['A','B','C','D']

представлено в памяти ЭВМ битовой строкой 1 1 1 1.

Подмножества этого множества представлены строками: ['A','B','D'] 1 1 0 1

['B','C'] 0 1 1 0

['D'] 0 0 0 1

Величины множественного типа не могут быть элементами списка ввода - вывода.

В каждой конкретной реализации транслятора с языка ПАСКАЛЬ количество элементов базового типа, на котором строится множество, ограничено. В TURBO PASCAL количество базовых элементов не должно превышать 256.

Инициализация величин множественного типа производится с помощью типизированных констант:

```
const seLit: Set of 'A'..'D' = [];
```

Проиллюстрируем применение данных множественного типа на примере.

Пример. Составить программу, которая вырабатывает и выводит на экран дисплея наборы случайных чисел для игры в "Спортлото 5 из 36".

Для заполнения каждой карточки спортлото необходимо получить набор из пяти псевдо-случайных чисел. К этим числам предъявляются два требования:

числа должны находиться в диапазоне 1..36; числа не должны повторяться.

```
until (n in nb) and not (n in k); k:=k+[n];
```

```
end.
```

Пример

Пример1: Дан текст. Определить каких букв больше - гласных или согласных.

Этапы решения задачи:

1. Составим блок схему программы

Опишем подробнее блок "Подсчитываем количество гласных и согласных букв" Рассмотрим блок "Печатаем соответствующее сообщение"

Запишем блок-схему целиком

2. Переведем алгоритм на язык Паскаль program example1;

```
const glasn=['a','e','и','o','y','ы','э','ю','я'];
```

```
soglas=['б','в','г','д','ж','з','й','л','м',  
'н','р','к','п','с','т','ф','х','ц','ч','ш','щ'];
```

```
var
```

```
begin
```

```
write('Введите строку> '); readln(st); g:=0; s:=0;
```

```
for i:= 1 to length(st) do
```

```
if st[i] in glasn then inc(g) else if st[i] in soglas then inc(s); if g> s then
```

```
writeln('Гласных больше')
```

```
else if g< s then writeln('Согласных больше')
```

```
else writeln('Согласных и гласных букв поровну'); readln;
```

```
end.
```

Практические задания:

Примечание: Гласные буквы - а,е,и,о,у,ы,э,ю,я (ё обычно не входит в литерный тип); согласные

- все остальные буквы, кроме ь, ъ; звонкие согласные -

б,в,г,д,ж,з,й,л,м,н,р; глухие согласные - к,п,с,т,ф,х,ц,ч,ш,щ.

Задание 1.

Дан текст из строчных латинских букв, за которым следует точка. Напечатать:

- первые вхождения букв в текст, сохраняя их взаимный исходный порядок;

- все буквы, входящие в текст не менее двух раз;
- все буквы, входящие в текст по одному разу.

Задание 2.

Дана непустая последовательность слов из строчных русских букв; между соседними словами - запятая, за последним словом - точка. Напечатать в алфавитном порядке:

все гласные буквы, которые входят в каждое слово; все согласные буквы, которые не входят ни в одно слово;

все звонкие согласные буквы, которые входят хотя бы в одно слово; все глухие согласные буквы, которые не входят хотя бы в одно слово;

все согласные буквы, которые входят только в одно слово; все глухие согласные буквы, которые не входят только в одно слово;

все звонкие согласные буквы, которые входят более чем в одно слово; все гласные буквы, которые не входят более чем в одно слово;

все звонкие согласные буквы, которые входят в каждое нечетное слово и не входят ни в одно четное слово; все глухие согласные буквы, которые входят в каждое нечетное слово и не входят хотя бы в одно четное слово.

Сформировать новое множество .

Предусмотреть формирование исходных множеств с клавиатуры.

Задание 4. Подсчитать общее количество цифр и знаков '+', '-', и '*', входящих в строку s.

Файлы последовательного доступа

Цель:

☐ изучение файловых типов данных, приобретение практических навыков создания и обработки файлов.

Краткие теоретические сведения

Работа с файлами

Информация, вводимая с клавиатуры или обрабатываемая с помощью программных средств Бейсика, размещается в оперативной памяти компьютера.

Алгоритм, набранный в Бейсике, может быть сохранен на диске в виде файла.

Файл — это поименованная область на магнитном или лазерном диске. В файлах могут содержаться тексты, графические и видеоизображения, звуки и музыка, таблицы и базы, данные программы, данные для этих программ.

Требования к имени файла

- имя не должно быть больше чем 8 символов;
- имя может состоять из букв латинского алфавита, цифр и символов, например, _, -, (,), \$ и некоторых других.
- в имени файла **запрещены** символы <Пробел>, *, точка, запятая, кавычки, двоеточие. Впрочем, злоупотреблять специальными символами не стоит — букв и цифр вполне хватает.

Расширение файла

Файл имеет расширение.

Расширение имени файла (англ. *filename extension*, часто говорят просто расширение файла или расширение) — последовательность символов, добавляемых к имени файла и предназначенных для идентификации типа (формата) файла. Расширение имеет длину не более трех символов, указывается через точку после имени.

Расширение файла	Описание формата файла
*.aif, *.aifc, *.aiff	Файлы аудиоданных
*.asm	Исходный текст программы на Ассемблере.
*.avi	Основной формат видеоизображений
*.bas	Текст программы на языке алгоритмическом языке Basic и различных его вариантах (GWBasic, TurboBasic, QuickBasic)
*.bmp	Формат графических файлов (растровая графика).
*.com	Исполняемый файл в двоичном коде
*.cpp	Текст программы на языке C++
*.doc	Файл с документами или продукт работы текстового процессора Microsoft Word for Windows
*.dot	Шаблон документа текстового процессора MS Word
*.exe	Это всегда исполняемый бинарный файл
*.gif	(Graphics Interchange Format). Растровый графический формат фирмы CompuServe
*.htm, *.html	Специальный файл текстового типа, написанный на Hyper Text Markup Language
*.mdb	Файл баз данных Microsoft Access
*.mov -	Формат хранения видео и аудио
*.pas -	Текст программы на языке Pascal

*.ppt	Файл с презентацией Microsoft PowerPoint
*.sys -	Системные файлы ядра DOS IO.sys и MSDOS.sys.
*.txt	Текстовый файл, созданный в блокноте
*.xls	Файл работы табличного процессора Microsoft Excel
*.zip -	Файл архива сжатого архиваторами

В файлах вы можете хранить как исходные данные для обработки, так и результаты работы программы.

Для работы в Бейсике необходимы файлы, хранящие однородные по типу или структуре сведения, о каких-либо объектах. Набор данных о каком-либо одном объекте называется **записью**.

Файл может быть пустым, т. е. содержать 0 байт информации, но имя файла и символ конца файла будут присутствовать. (**Байт** - единица измерения количества информации, объема памяти и емкости запоминающего устройства. По умолчанию байт считается равным 8 битам).

Записи могут содержать данные разных типов, но должны быть обязательно одинаковы по структуре, например:

"Запорожец", "4067 ЛДЕ", "1972", "100\$"

"ГАЗ-34", "6666 ЛАА", 1989, "3500\$"

В соответствии со способом доступа к файлам они делятся на два вида.

1. Файл с последовательным доступом;
2. Файл с прямым доступом.

Файлы последовательного доступа наиболее просты как в организации, так и в работе с ними. Записи обрабатываются последовательно одна за другой.

Информация в таких файлах хранится в виде текста в кодах ASCII. Такие файлы легко просмотреть на экране, используя любой простейший редактор, или в самом Бейсике.

Простота — хорошо, а последовательность в данном случае — плохо. Если информация об интересующем объекте упорядочена в файле по алфавиту, то придется перебирать практически весь файл, чтобы добраться до нужной записи. Отсюда, при большом информационном объеме файла обработка его замедляется.

Файлы прямого доступа хранят информацию в специальном формате, в котором каждая запись занимает строго фиксированную одинаковую с остальными длину. Такие файлы занимают на диске больше места, чем файлы последовательного доступа, но скорость работы с ними значительно выше.

Операции над файлами

Независимо от того, какие действия происходят с информацией, хранящейся в файле, производятся следующие обязательные операции:

1. открытие файла;
2. чтение и запись обрабатываемых данных;

В Паскале определены текстовые файлы, типизированные и нетипизированные. Файл, не содержащий ни одного элемента, называется пустым. Создается файл путем добавления новых записей в конец первоначально пустого файла. Длина файла, т.е. количество элементов, не задается при определении файла.

Все файлы должны быть описаны в программе либо в разделе переменных VAR, либо в разделе типов TYPE. Под чтением файла понимают ввод данных из внешнего файла, находящегося на диске, в оперативную память машины. Запись в файл - вывод результатов работы программы из оперативной памяти на диск в файл.

Работа с файлами выполняется следующими процедурами:

Assign – устанавливает связь между именем файла в программе (файловой переменной) и физическим именем файла, принятым в ОС.

Reset - открывает существующий файл для чтения.

Rewrite – создает и открывает новый файл для записи на внешнем устройстве (если файл ранее существовал, вся предыдущая информация из него стирается).

Close - закрывает открытый файл.

Для определения конца файла используется стандартная встроенная функция EOF (файловая переменная), которая принимает значение True, если достигнут конец файла, и значение False в противном случае.

2.1. Текстовые файлы

Текстовые файлы – файлы на диске, состоящие из символов ASCII. Для разделения строк используются символы «конец строки». Текстовые файлы являются файлами с последовательным доступом. В любой момент времени доступна только одна запись файла. Другие записи становятся доступными лишь в результате последовательного продвижения по файлу. Текстовые файлы внутренне разделены на строки, длины которых различны. Для разделения строк используется специальный маркер конца строки. Объявляются текстовые файлы переменной типа text. Обработать их можно только последовательно и с помощью процедур и функций:

Readln (f , st)- чтение строки st из файла f и переход на начало следующей ;
Writeln (f, st)- запись строки st в файл f и маркера конца строки ;

Append (f) - процедура, открывающая файл f для добавления строк в конец файла;

Eoln (st)- логическая функция, результат выполнения которой равен TRUE, если достигнут маркер конца строки st.

Пример 1. Создать текстовый файл, в который записать 3 предложения. Прочитать этот файл, вывести его содержимое на экран. Определить длину каждого предложения.

begin

assign (f1, 'file1.txt'); {связать с файлом file1.txt файловую переменную f1 }
rewrite (f1); { создать новый файл с именем file1.txt }

```

        writeln ( f1, 'Очень полезно изучать'); { записать предложения в файл} writeln (
f1, ' всем студентам ');
        writeln (f1, ' язык Pascal ');
        close (f1);           { закрыть файл для записи }
        reset (f1); { открыть файл для чтения }
        while not eof (f1) do { пока не конец файла f1 } begin
            readln (f1, st);           {читаем строку из файла f1 } writeln(st); {
выводим на экран }
            n:= length (st);           {определяем длину строки } writeln (' длина
=' ,n);
        end;
        close (f1); { закрыть файл для чтения} end .

```

Практические задания

Задание

1. Разобрать и проанализировать приведенные программы.
2. Используя подпрограммы, создать внешний файл из 7

записей, прочитать создан-ный файл и, применяя режим прямого доступа, выполнить задания по своему варианту.

Типизированные файлы

Цель:

☐ изучение файловых типов данных, приобретение практических навыков создания и обра-ботки файлов.

Краткие теоретические сведения Типизированные файлы

Типизированные файлы – это файлы, состоящие из нумерованной последовательности объектов (записей) любого типа. С такими файлами можно работать в режиме прямого доступа, при котором выполняется непосредственное обращение к любой записи файла. Каждая запись файла имеет свой номер, начиная с 0 и т.д.

Процедуры и функции обработки файлов:

- 1) Write и Read- записывают и читают информацию из указанного файла и переме-щают указатель файла к следующей записи.
- 2) Seek (файловая переменная, номер записи); процедура перемещения указателя на запись файла с заданным номером .
- 3) Truncate (файловая переменная); процедура, усекающая файл по текущей позиции указателя файла, т.е. все записи, находящиеся после указателя файла, удаляются.
- 4) Функция Filesize (файловая переменная); имеет тип Integer и определяет размер файла, т.е. число записей.

5) Функция Filepos (файловая переменная); имеет тип Integer и возвращает текущую позицию указателя файла.

Для добавления записей в конец файла используются процедуры: Readln (a);

Seek (f, filesize (f));

Write (f, a);

При этом указатель устанавливается за конец файла, т.к. нумерация записей начинается с нуля. После чего с помощью Write можно добавлять записи. Открывать файл можно только процедурой Reset (f).

Для того, чтобы в режиме произвольного доступа считать, а затем изменить значение за-писи, следует выполнить два вызова процедуры Seek. Один вызов перед операцией Read, а другой

- перед операцией Write (т.к. Read после чтения записи переместит указатель к следующей за-писи).

Пример: Создать файл из списка 10 студентов с их оценками (номер, Ф.И.О. и три оценки). Вывести его содержимое на экран, изменить фамилию студента с номером, введенным с клавиатуры, заново прочитать файл.

Program lab6;

Type

wed = record {Тип wed включает 3 поля: n, fio, bal} n : byte ; fio : string[15] ;

bal : array [1..3] of byte; {Поле bal – массив из 3 оценок } end;

Var spisok : wed ; {Запись spisok типа wed} sp : file of wed; {Файл записей типа wed}

procedure vvod; { процедура создания файла} var i,j:byte;

begin

{ оператор assing находится в основной программе } rewrite (sp); {открытие файла для записи} with spisok do

For i:=1 to 10 do begin n:=i;

writeln (' Введите фамилию - ', i); readln (fio);

writeln (' Введите 3 оценки ', fio); For j:= 1 to 3 do readln (bal [j]);

write (sp , spisok); { запись в файл информации о студенте}

end;

close (sp); { закрытие файла для записи } end;

procedure print; { процедура чтения и печати всего файла } var j : byte;

begin

reset (sp);

Практические задания

Задание

1. Запись имеет вид: фамилия, пол, год рождения и рост. Вывести данные о самом высоком спортсмене.

2. Запись имеет вид: название вуза, число студентов, количество

факультетов. Добавить в конец файла информацию о трех новых вузах.

3. Запись имеет вид: название издания, газеты или журнала, стоимость одного экземпляра, количество экземпляров в год . Вывести на печать информацию о самом дешевом издании.

4. Запись имеет вид: фамилия студента, номер зачетной книжки, 4 оценки за экзамен. Вы-водить информацию о всех двоечниках и корректировать ее.

5. Запись имеет вид: фамилия спортсмена, его номер, количество набранных очков. По-менять местами в файле записи о первых двух спортсменах. *

6. Запись имеет вид: фамилия, номер телефона, дата рождения. Внести в начало списка информацию о четырех новых знакомых.

Применение рекурсивных функций

Цель:

– научиться решать задачи с использованием рекурсивных функций и процедур

Краткие теоретические сведения

Если поставить два зеркала напротив друг друга и между ними поместить предмет, то по-лучится бесконечное множество изображений, причем каждое из них содержит свое собственное. Любое из этих изображений можно рассматривать как рекурсивный объект, который частично состоит или определяется с помощью самого себя. Рекурсивные объекты обладают несколькими свойствами: — простотой построения;

- несхожестью конечного результата с начальными данными;
- внутренним самоподобием.

В математике встречаются рекурсивные определения, позволяющие описать объекты че-рез самих себя. К таким определениям относится, например, определение натурального числа:

- 1) единица есть натуральное число;
- 2) число, следующее за натуральным (т. е. больше его на единицу), есть натуральное число.

Определение, которое задает некоторый объект в терминах более простого случая этого

же объекта, называется рекурсивным определением.

Мощность рекурсивного определения заключается в том, что оно позволяет с помощью конечного высказывания определить бесконечное множество объектов. Как и цикл, рекурсивное определение содержит повторения, но каждый раз при этом используются новые данные, т. е. повторения не являются явными.

Рекурсия — это способ описания функций или процессов через самих себя.

Процесс может быть описан некоторым алгоритмом, называемым в данном случае рекур-сивным. В нем выделяется два этапа выполнения:

1) «погружение» алгоритма в себя, т. е. применение определения «в обратную сторону», пока не будет найдено начальное определение, не являющееся рекурсивным;

2) последовательное построение от начального определения до определения с введенным в алгоритм значением.

Примеры рекурсивных алгоритмов, часто оформляемых в виде процедур и функций.

1. Наиболее распространенным рекурсивным определением является определение факториала (нерекурсивное вычисление факториала приведено в примере Р9): $1! = 1$,
 $n! = n * (n - 1)!$

На основе этого определения можно записать программу вычисления факториала, использующую рекурсивную функцию.

program P25;

var n, y: **integer**;

function F (x: **integer**): **integer**; {описание рекурсивной функции} **begin**

if x=1

then F := 1 {вызов для начального определения}

else F := x * F (x - 1) {вызов для предыдущего определения}

end; {конец описания функции} **begin** {начало главной программы}

readln (n);

Y := F (n); {вызов функции в главной программе} **write** (n, '!' = ', Y)

end. {конец главной программы}

Выполним программу P25 для $n = 4$. Рекурсивная функция будет работать следующим образом (при вызове функции значение n присваивается переменной x). Сначала осуществляется

«погружение», работает оператор ветви **else** условного оператора:

1- й шаг: $x = 4$, $x-1 = 3$, выполняется промежуточное вычисление $4! = 4 * 3!$

2- й шаг: $x = 3$, $x-1 = 2$, выполняется промежуточное вычисление $3! = 3 * 2!$

3- й шаг: $x = 2$, $x-1 = 1$, выполняется промежуточное вычисление $2! = 2 * 1!$

4- й шаг (последний): $1! = 1$ по начальному определению, работает оператор F := 1 ветви

then условного оператора.

Следующий этап выполнения рекурсивного алгоритма — построение «прямого» определения, от начального до получения результата с исходными для алгоритма данными (числом 4). При этом осуществляется подстановка предыдущих вычислений (более поздних шагов) в более ранние:

5- й шаг: $2! = 2 * 1 = 2$

6- й шаг: $3! = 3 * 2 = 6$

7- й шаг: $4! = 4 * 6 = 24$ — получен результат, он возвращается в главную программу и присваивается переменной Y.

Практические задания

Задание.

№	Условие задачи
1	Написать функцию, которая находит цифровой корень целого числа.
2	Найти сумму цифр заданного натурального числа.
3	Найти количество цифр в заданном натуральном числе.
4	Составить программу вычисления суммы четных факториалов. (n-четное, $n \leq 10$)
5	Описать рекурсивную логическую функцию $Simm(S, I, J)$, проверяющую, является ли симметричной часть строки S, начинающаяся i-м и заканчивающаяся j-м ее элементом.
6	Составить программу вычисления суммы нечетных факториалов. (n-нечетное, $n \leq 10$)
7	Составить программу сортировки массива целых чисел.
8	Составить программу вычисления НОД двух натуральных чисел.
9	Составить программу нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке. (173 → 371)
10	Составить программу перевода данного натурального числа в r-ичную систему счисления ($2 \leq r \leq 9$)
11	Дан прямоугольник со сторонами A и B, где A, B- натуральные числа. Начнем отсекать от него квадраты. Сколько квадратов можно отсечь, если каждый раз отсекается самый большой квадрат.
12	Поиска значений в упорядоченном списке.
13	Найти сумму $1/1 + 1/2 + 1/3 + 1/4 + \dots + 1/n$, основываясь на рекурсии. (сумма k слагаемых равна сумме (k-1) слагаемых плюс k-е слагаемое).
14	Напишите главную программу для вычисления n-го числа Фибоначчи. Почему использовать рекурсивный алгоритм вычисления n-го числа Фибоначчи невыгодно?
15	Определите рекурсивно умножение как сложение и деление как вычитание и оформите алгоритмы в виде рекурсивных функций с вызовом из главных программ.
16	Составить программу сортировки массива целых чисел.
17	Составить программу вычисления НОД двух натуральных чисел.

18	Составить программу нахождения числа, которое образуется из данного натурального числа при записи его цифр в обратном порядке.(173□371)
19	Составить программу перевода данного натурального числа в р-ичную систему счисления(2□p□9)
20	Дан прямоугольник со сторонами А и В, где А,В- натуральные числа. Начнем отсекать от него квадраты. Сколько квадратов можно отсечь, если каждый раз отсекается самый большой квадрат.
21	Поиска значений в упорядоченном списке.
22	Найти сумму $1/1+1/2+1/3+1/4+...+1/n$,основываясь на рекурсии.(сумма k слагаемых равна сумме (k-1) слагаемых плюс k-е слагаемое).

Практическая работа №25-51.

Программирование модуля

Цель:

– усвоить знание основ модульного программирования; освоить способы создания и приме-нения модулей.

Краткие теоретические сведения

Модульное программирование основано на понятии **модуля** – **программы** или функцио-нально завершенного **фрагмента** программы.

Модуль характеризуют:

- 1) один вход и один выход. На входе программный модуль получает определенный набор исходных данных, выполняет их обработку и возвращает один набор выходных данных;
- 2) функциональная завершенность. Модуль выполняет набор определенных операций для реализации каждой отдельной функции, достаточных для завершения начатой обработки дан-ных;
- 3) логическая независимость. Результат работы данного фрагмента программы не зависит от работы других модулей;
- 4) слабые информационные связи с другими программными модулями. Обмен информа-цией между отдельными модулями должен быть минимален;
- 5) размер и сложность программного элемента в разумных рамках.

С помощью модулей решаются различные профессиональные задачи обработки данных разного типа.

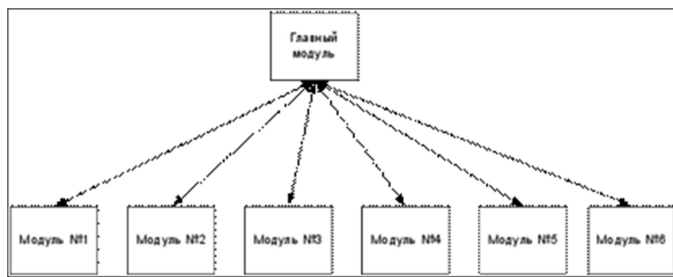


Рис. 1. Схема подключения модулей

Предлагается использовать следующие **характеристики программного модуля** для оценки его приемлемости: **размер** модуля, **прочность** модуля, **сцепление** с другими модулями и **рутинность** модуля.

Размер модуля измеряется числом содержащихся в нем операторов. Модуль не должен быть слишком маленьким или слишком большим.

Прочность модуля – это мера его внутренних связей. Чем выше прочность модуля, тем больше связей скрыто от внешней по отношению к нему части программы и, следовательно, тем проще сама программа.

Сцепление модуля – это мера его зависимости по способу передачи данных от других модулей. Чем слабее сцепление модуля с другими модулями, тем сильнее его независимость от других модулей.

Рутинность модуля — это его независимость от предыстории обращений к нему. Модуль будем называть рутинным, если результат обращения к нему зависит только от значений его параметров и не зависит от результатов предыдущих обращений к нему.

С помощью модулей решаются различные профессиональные задачи обработки данных разного типа.

Практические задания

Задача 1. Разработать и реализовать программу ввода данных, вычисления функций при помощи модулей, вывода результатов.

Программа работы

1. Составить список функций и соответствующих им модулей для реализации в программе.
2. Разработать интерфейс программы ввода, обработки и вывода данных.
3. Написать программный код для каждого модуля.
4. Реализовать интерфейс и программный код в среде визуальной разработки программы.
5. Провести тестирование и отладку программы.
6. Нарисовать интерфейс программы со спецификацией и записать программный код с комментариями в отчете по работе.

7. Записать несколько вариантов тестирования программы.
8. Провести тестирование исполняемого файла.

Задача 2. Разработать и реализовать программу вычисления значения сложного выражения при помощи шагов:

1. Ввода данных, вычисления функций при помощи первого модуля, вывода результатов;
2. Чтения выходных данных первого модуля, вычисления функций при помощи второго модуля, вывода результатов;
3. Чтения выходных данных второго модуля, вычисления функций при помощи третьего модуля, вывода результатов.

Программа работы

1. Составить список функций и соответствующих им модулей для реализации в программе.
2. Разработать интерфейс программы ввода, обработки и вывода данных.
3. Написать программный код для каждого модуля.
4. Реализовать интерфейс и программный код в среде визуальной разработки программ.
5. Провести тестирование и отладку программы.
6. Нарисовать интерфейс программы со спецификацией и записать программный код с комментариями в отчете по работе.
7. Записать несколько вариантов тестирования программы.
8. Провести тестирование исполняемого файла.

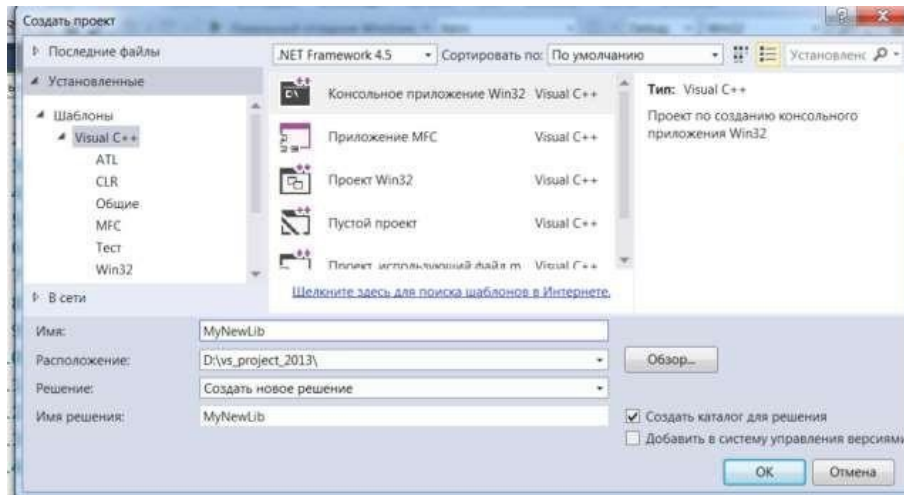
Использование указателей для организации связанных списков Цель:

- изучение порядка действий при вычислении выражений;
- приобретение навыков в записи выражений на языке Паскаль и использовании стандартных функций.

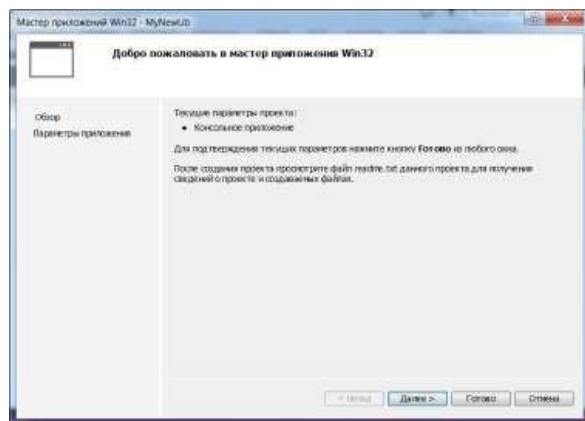
Цель работы: сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с использованием библиотеки подпрограмм **Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

Существует два вида библиотек статические .lib и динамические .dll,

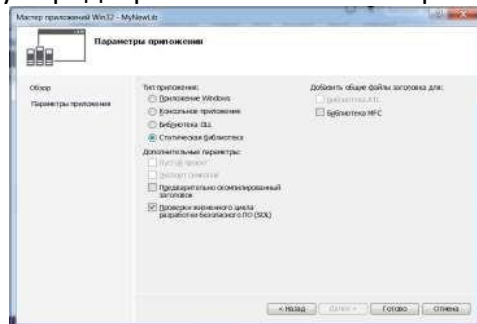
1. Открываем visual studio у меня visual studio
2. Выбираем: "Создать новый проект", в открывшемся окне выбираем "консольное приложение виндовс 32", также вводим название проекта MyNewLib и указываем путь где мы сохраним проект.



Дальше нажимаем ок и мы переходим к следующему окошку.



3. В новом окне выбираем “Статическая библиотека” и убираем галочку “предварительнооткомпилированные заголовки”, далее нажимаем «готово»



Таким образом, у нас создается новый проект в котором будут создаваться не программы exe астатические библиотеки lib.

4. Дальше в этот проект MyNewLib добавляем два файла один заголовочный

```
#ifndef FUNCTION_H#define FUNCTION_H

void func_hellow();void func_privet();
```

function.h:

и файл function.cpp:

```
#include <iostream>using std::cout; using std::endl;

#include "function.h";

void func_hellow()
{
    cout <<"Hellow world gacpada"<<endl;
}

void func_privet()
{
```

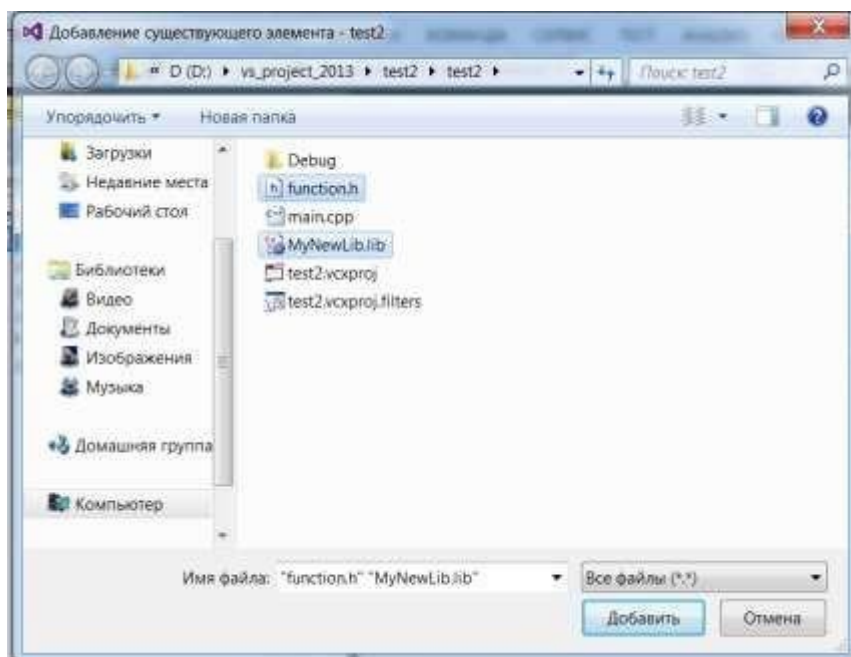
Включаем их в проект, а дальше компилируем нажимаем клавиши ctrl+alt+F7 и у нас создается в папке

Debug или Release смотря какой режим выбрать. (мы выберем режим Debug) и в папке Debug создан файл библиотеки MyNewLib.lib

Для того, чтобы создавать качественные библиотеки нам нужно каждую функцию записать в отдельный файл, это для того, чтобы в MyNewLib.lib было создано несколько объектных файлов .obj в данном случае мы могли бы файл function.cpp разбить на два файла допустим файл f1.cpp в котором записать определение функции func_hellow() и на файл f2.cpp в нем записать определение второй функции func_privet(), тогда у нас будет правильная библиотека и если мы в проекте будем использовать одну функцию, допустим func_hellow(), а func_privet() нет, то объектный код .obj с функцией func_privet() не добавится, а только

.obj func_hellow() будет добавляться, я так думаю это важный момент, возможно ниже попробуем смоделировать два разных варианта создания библиотек. Просто при использовании библиотеки в которой каждой функции принадлежит отдельный объектный код, только включаются те функции которые используются, а если все функции в одном .obj то естественно включаются все функции и даже те которые мы не используем.

5. Нужно подключить библиотеку к нашему проекту, для этого:



Выбираете файлы и нажимаете добавить и все файлы добавлены, при открытии обозревателя решений у вас они должны быть видны из него.

Все из рисунка видно что статическая библиотека MyNewLib.lib у нас добавлена и заголовочный файл с определениями функций из библиотеки MyNewLib.lib включен это файл function.h, теперь нажимаем Ctrl+F5 и все компилируется.

Создание проекта с использованием компонентов для работы с текстом

Цель работы: сформировать умения по использованию и настройке интегрированной среды разработчика Delphi, сформировать умения по созданию простейших приложений

Оборудование, технические и программные средства: персональный компьютер, интегрированная среда разработчика **Turbo Delphi**.

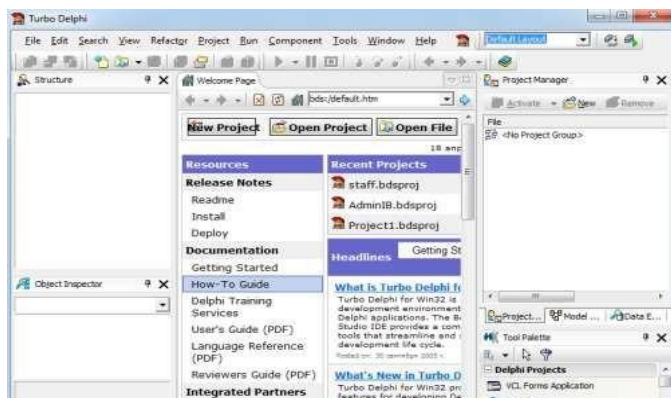
Задание 1. Изучение рабочей среды Turbo Delphi Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Для запуска **Turbo Delphi** существует несколько способов:

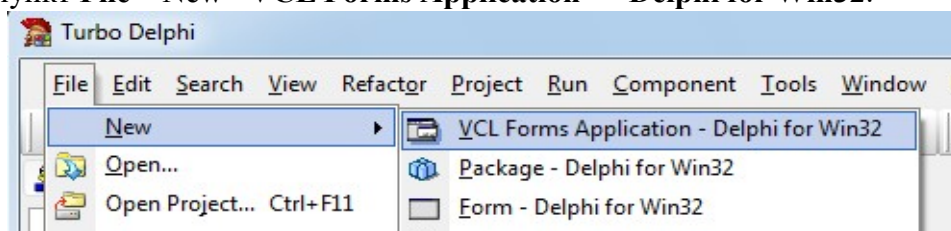
- Пуск – Все Программы – Borland Developer Studio 2006 – Turbo Delphi;
- Двойной щелчок на пиктограмме программы в окне **Компьютер**;
- Двойной щелчок на соответствующем ярлыке на рабочем столе.

Turbo Delphi — это среда быстрой разработки от корпорации **Borland**, в которой в качестве языка программирования используется одноименный язык программирования **Delphi**, ранее из-вестный как **Object Pascal**.

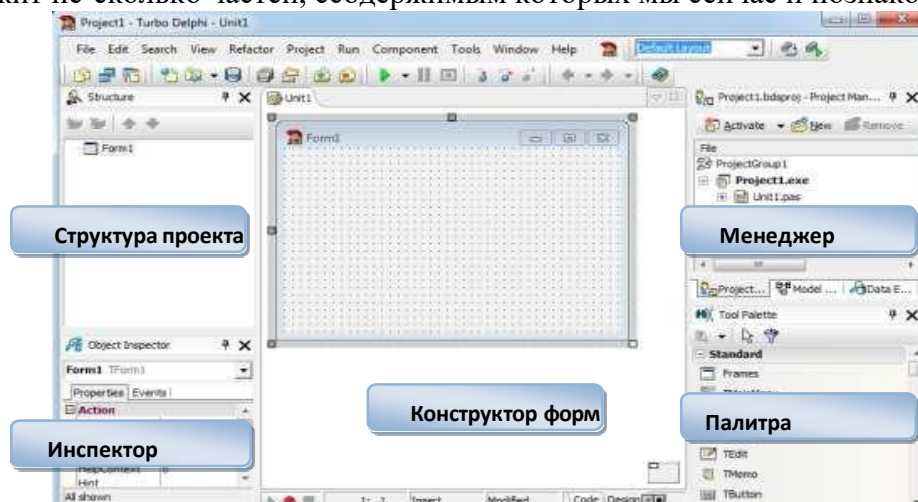
2. После запуска появится главное окно среды разработки в центре которого по умолчанию будет открыта HTML-страница **Welcome Page** во встроенном Интернет - браузере. На ней находятся ссылки на разделы документации и список последних открытых проектов (**Recent Projects**).



3. Закройте страницу приветствия и создайте новый проект, выбрав в главном меню пункт **File – New - VCL Forms Application — Delphi for Win32**.



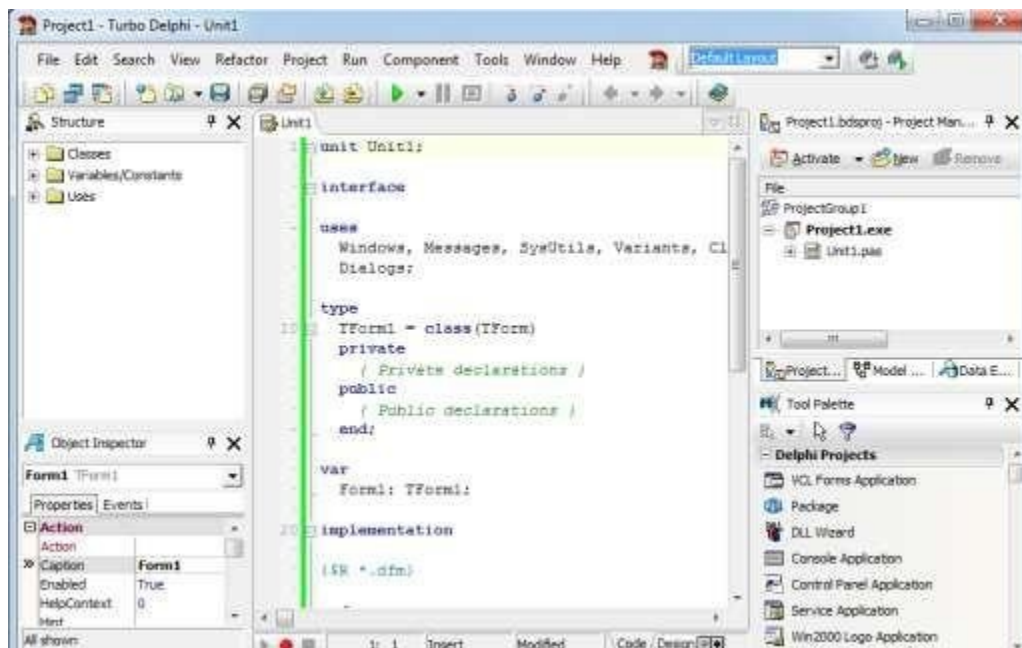
4. Среда разработки примет вид, представленный на рисунке. Данное окно содержит не-сколько частей, содержанием которых мы сейчас и познакомимся.



5. Центральную часть окна занимает окно **Конструктора форм (Form designer)** приложения. **Формой**

(Form) приложения на этапе разработки принято называть окно программы (во время разработки это форма, на этапе выполнения — окно). В окне конструктора форм вы будете формировать внешний вид своего будущего приложения: изменять само окно, а также наполнять его различными элементами. Окно конструктора формы изначально находится на переднем плане и перекрывает окно редактора кода. Переключаться между этими окнами можно нажатием клавиши **F12** (либо нажатием на специальные вкладки **Code** и **Design** в нижней части окна).

6. Активизируйте окно редактора кода, нажав клавишу **F12** на клавиатуре. Содержимое окна программы будет выглядеть так, как показано на рисунке.



Необходимо отметить, что большая часть работы при создании проекта в среде **Delphi** сводится к работе конструктором форм или редактором кода, при этом посредством конструктора создается внешнее окно приложения, а в редакторе прописывается код программы.

При закрытии конструктора форм или редактора кода автоматически закрывается и приложение. Вернитесь обратно к окну **Конструктора формы**.

7. Слева от окна конструктора находится окно **Инспектора объектов (Object Inspector)**. Это окно заполнено информацией о выделенном объекте проекта (в данный момент — формы). Окно **Object Inspector** имеет две вкладки — **Properties (Свойства)** и **Events (События)**.

8. Измените для формы свойство **Caption**. Это свойство отвечает за заголовок формы. Изначально оно равно **Form1**, измените его на **Мой первый проект на Delphi** и нажмите клавишу **Enter**, сразу увидите, что ваша форма изменилась. Во время выполнения окна вашего приложения будет иметь введенный вами заголовок.

9. Поначалу группировка свойств может вызвать затруднения (свойств довольно много у любого объекта и для запоминания основных из них нужно некоторое время), поэтому такую группировку можно отключить. Для этого необходимо навести указатель мыши на окно **Object Inspector** и нажать правую кнопку мыши, в появившемся вспомогательном меню выбрать пункт **Arrange - by Name**.

Задание 2. Создание простого проекта

Разработать программу **Hello**, которая при нажатии на кнопку «**Hello**» выводит сообщение «**Hello, мир Windows and Delphi!**», при нажатии на кнопку «**Выход**» программа завершит работу.

Методические указания по выполнению задания:

1. **Проект (Project)** — совокупность файлов, используемые средой разработки (точнее говоря, компилятором **Turbo Delphi**) для итоговой генерации программы. Проект в простейшем случае представляет собой совокупность следующих файлов:

- **файл описания проекта (bdsproj-файл)** — файл специального формата, в котором записана общая информация о проекте;

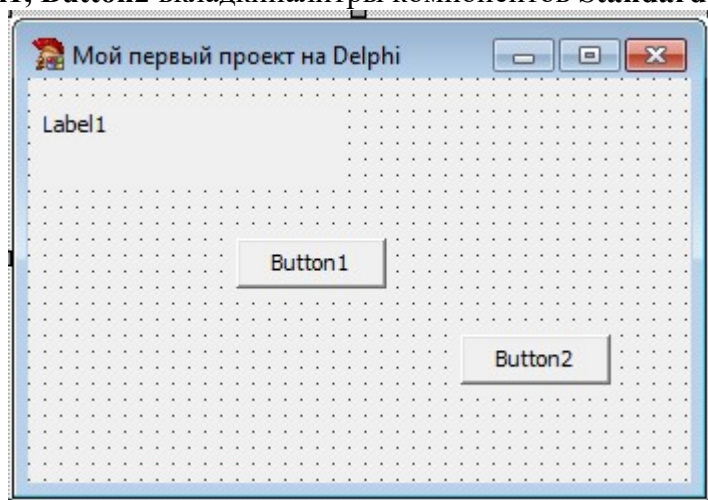
- **главный модуль (dpr-файл)** содержит инструкции, обеспечивающие запуск нашей программы;
- **модуль формы (dfm-файл)** содержит информацию о настройках и компонентах, которые присутствуют на форме. Модуль формы формируется автоматически при выполнении настроек формы, перенесении на нее и настройки компонентов;
- **модуль реализации (pas-файл)** содержит информацию только о присутствующих на форме компонентах и процедурах обработки событий на этих компонентах;
- **файл ресурсов (res-файл);**
- **файл конфигурации (cfg-файл);**
- **исполняемый файл (exe-файл)**, который создается при запуске программы на исполнение.

Поскольку при работе с проектом автоматически создается довольно много файлов, рекомендуется сохранять их в заранее подготовленном каталоге. Создайте в папке своей группы новую папку и назовите её **Hello**.

2. Сохранение всех файлов проекта осуществляется через пункт главного меню **File - Save all (сохранить все)**. Первый раз при сохранении потребуются сохранить два файла: модуль формы и файл проекта. Сохраните проект, выполнив команду **File - Save Project as...** В открывшемся окне найдите свою новую папку (**Hello**). В этом же окне **Delphi** предложит сохранить (по умолчанию) первый файл под именем **Unit1.pas**. Можно согласиться, т.е. нажать кнопку **Сохранить**. Сразу же **Delphi** предложит сохранить второй файл под именем **Project1.bdsproj**. Его надо обязательно переименовать в **hello.bdsproj**

3. Откройте папку **Hello**, где вы сохранили файлы. Обратите внимание, что на самом деле вы создали более двух файлов, все они и образуют проект. Из этих файлов основным рабочим файлом является файл с расширением **.pas** (в нашем примере **Unit1.pas**)

4. Разместите на форме компоненты в соответствии с рисунком: **Label1**, **Button1**, **Button2** вкладки палитры компонентов **Standard**.



5. Выделите кнопку **Button2**, перейдите в **Object Inspector** на вкладку **Properties**, найдите свойство **Caption (Заголовок)** и измените его на **Выход**.

6. Сохраните проект, нажав на соответствующую кнопку на панели инструментов.

7. Перейдите на правую страницу **Object Inspector**, на страницу **Events**, и найдите событие **OnClick**, справа от него дважды щелкните мышью.

8. Из листинга видно, что в рамках формы класса **TForm1** описываются все

присутствующие на форме компоненты: **Label11**, **Button1**, **Button2** и заголовок процедуры **Button2Click**. В реализационной части, после служебного слова **implementation**, создана заготовка для всей процедуры, в которой и будем вписывать необходимый код. Параметр **Sender** передается в любой процедуре обработки события и определяет компонент формы, где произошло событие. Оказавшись в коде программы, а точнее, в заготовке процедуры кнопки **Button2**, напишите лишь одну команду: **Close**.

Все события, на которые приложение может реагировать, разделяются на пользовательские и системные. К пользовательским относятся события, связанные с клавиатурой или мышью, например **OnClick** — одинарный щелчок левой клавишей мыши (именно это событие является наиболее распространенным и именно для его обработки нами была создана процедура в предыдущем примере); **OnDblClick** — двойной щелчок левой клавишей мыши; **OnMouseDown** — нажатие клавиши мыши; **OnMouseUp** — отпускание клавиши мыши; **OnMouseMove** — перемещение мыши.

Кроме пользовательских событий, существуют программно управляемые события. Отдельно рассмотрим события, обрабатываемые самой формой: событие **OnCreate** происходит в момент создания формы; **OnClose** генерируется, когда форма должна быть закрыта. Два данных события происходят с формой всего один раз в отличие от других: **OnShow** — возникает, когда форма должна стать видимой, **OnHide**

— когда форма должна быть убрана с экрана, **OnPaint** — перед тем, как форма будет перерисована на экране.

Существуют события, которые формируют сами элементы управления, например: **OnEnter** событие, которое появляется, когда элемент управления становится активным, и **OnExit** событие, возникающее, когда элемент управления перестает быть активным. Данные два события существуют только у элементов управления и только в том случае, если свойство **Enabled** имеет значение **True**.

9. Выделите кнопку **Button1**, найдите в **Object Inspector** свойство **Caption** и вставьте новое название кнопки **Hello**.

10. Перейдите в **Object Inspector** на страницу **Events**, найдите событие **OnClick** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
  Label1.Caption := 'Hello, мир Windows and Delphi';  
end;
```

кнопки **Button1**, напишите следующее выражение:

11. Сохраните проект еще раз, запустите и протестируйте его, т.е. запустите проект и нажмите кнопку **Hello**

— должна появиться ранее написанная вами фраза, нажмите кнопку **Выход**, и программа закроется.

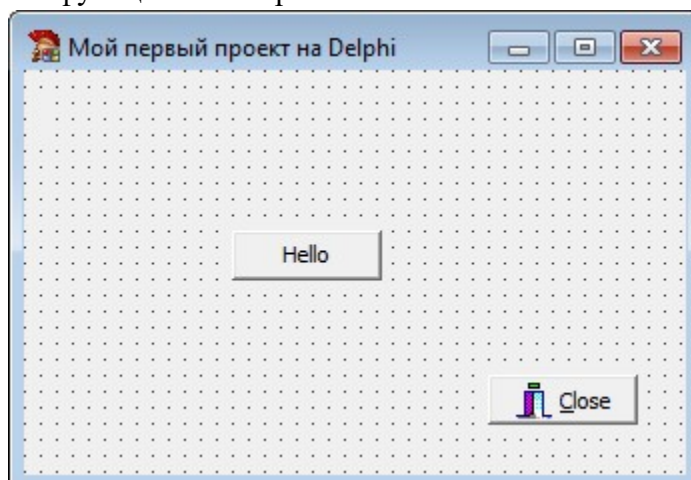
12. Измените содержание выводимой на экран реплики на «**Это моя первая программа**». Для этого дважды щелкните по кнопке **Hello** и измените соответствующий оператор.

13. Задайте шрифт и размер выводимой реплики следующим образом: размер — **16**, цвет — **красный**, гарнитура шрифта — **Arial**. Для этого измените свойство **Font** в **Object Inspector**.

14. Замените простую кнопку **Выход** на **Close**. Для замены кнопки надо

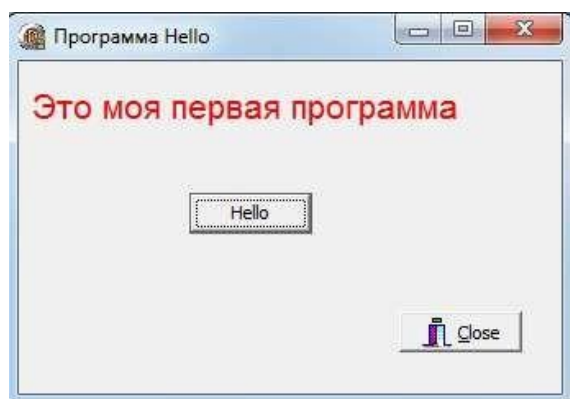
удалить существующую, найти в палитре компонентов на странице **Additional**, кнопку с названием **BitBtn**,

и разместить её на форме. Затем в свойстве **Kind** измените ее вид на **bkClose**. При этом в процедуре **Button2Click** можно убрать команду **Close**, т.к. кнопка этого типа (**BitBtn**) автоматически выполняет функцию по закрытию окна.



15. Запустите свой проект не в среде **Delphi**, а в среде **Windows**. Для этого сверните или закройте **Turbo Delphi**. Перейдите в папку вашего проекта **Hello** и найдите в ней файл **Hello.exe**. Щелкните мышкой по этому файлу – файл, как и положено файлу с расширением **.exe**, запустится. Вы увидите, что ваш проект работает в **Windows** без запуска среды **Delphi**.

16. Озаглавьте окно проекта «**Программа Hello**». Для этого выделите форму **Form1**, перейдите в **Object Inspector**, найдите свойство **Caption** и замените надпись на **Программа Hello**.



17. Сохраните изменения и запустите проект на исполнение.

Задание 3. Создание простого проекта

Разработать программу с помощью, которой пользователь, введя свой рост и фактический вес, мог бы определить худой он или полный, и на сколько ему нужно поправиться или похудеть. Для разработки программы воспользуйтесь тем, что оптимальный вес человека определяется так: рост человека минус 100.

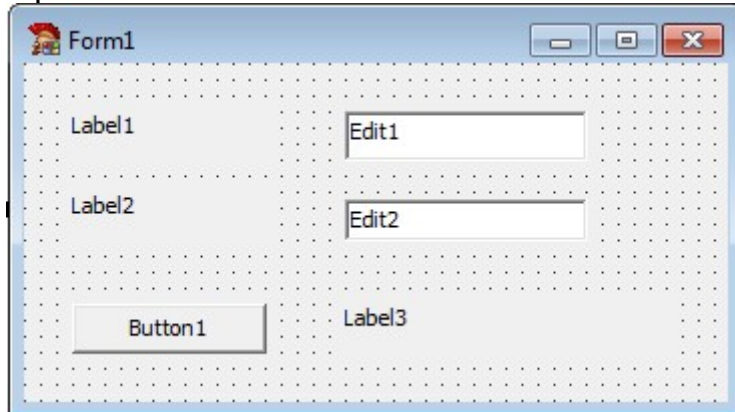
Если фактический вес человека меньше оптимального, то человек худой, и наоборот, если больше, то нужно похудеть.

Методические указания по выполнению задания:

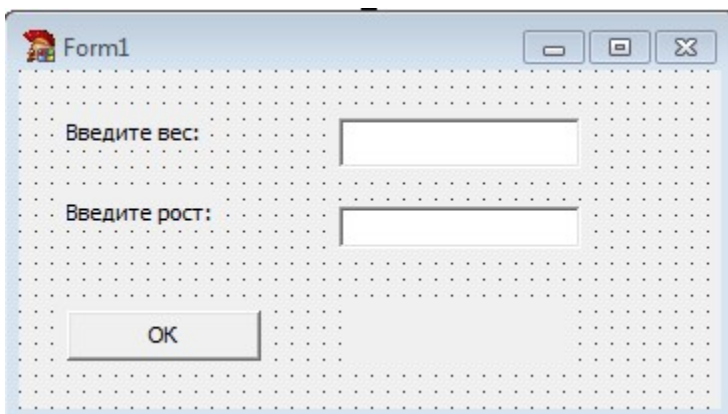
1. Создайте в папке своей группы новую папку и назовите её **Weight**.
2. Сохраните оба файла проекта в папке **Weight**, выполнив команду **File - Save**

Project as...

3. Разместите на форме компоненты в соответствии с рисунком. В **Edit1** будет вводиться вес в кг, а в **Edit2** – рост в см.



4. Задайте для элементов управления значение свойства **Caption** в соответствии с рисунком. Для **Label3** значение свойства **Caption** оставьте пустым, а для элементов управления **Edit1** и **Edit2** значение свойства **Text** оставьте пустым. Для этого выделите элемент управления и перейдите в **Object Inspector** на страницу **Properties (свойства)**.



5. Перейдите в код программы (на клавиатуре нажмите клавишу **F12**). Введите в раздел **VAR** переменные для сохранения значений фактического веса (**faktW**), оптимального веса (**optW**), роста (**Rost**) и разницы между оптимальным весом и фактическим (**Delta**). Пусть значения этих переменных будут округленными, в этом случае тип этих переменных можно объявить как **Integer**.

```
var
    Form1: TForm1;
    faktW, optW, rost, Delta: integer;
```

```

procedure TForm1.Button1Click(Sender: TObject);
begin
    faktW:= StrToInt(Edit1.text);
    Rost:= StrToInt(Edit2.Text);
    OptW:=Rost - 100;
    Delta:= abs (faktW - OptW);
    if OptW = faktW then Label3.caption := 'Ваш вес оптимален'
    else
    if OptW > faktW then Label3.caption := 'Вам надо поправиться на'
    +IntToStr(Delta)+' кг.'
    else
    Label3.caption := 'Вам надо похудеть на '+IntToStr(Delta)+' кг.'
end;

```

6. работу приложения.
7. Усовершенствуем программу так, чтобы можно было бы вводить десятичные величины. Для этого в разделе **Var** зададим тип переменных не **Integer**, а **Real**.
8. Функция **FloatToStr()** преобразует помещенную в скобки переменную типа **Real** в переменную строкового типа, а функция **StrToFloat()** превращает строку в дробное или действительное число, число с плавающей точкой. Замените в тексте программного кода **IntToStr** на **FloatToStr**, а **StrToInt** на **StrToFloat**.
9. Сохраните проект и протестируйте работу приложения.
10. Для форматирования и преобразования действительных чисел

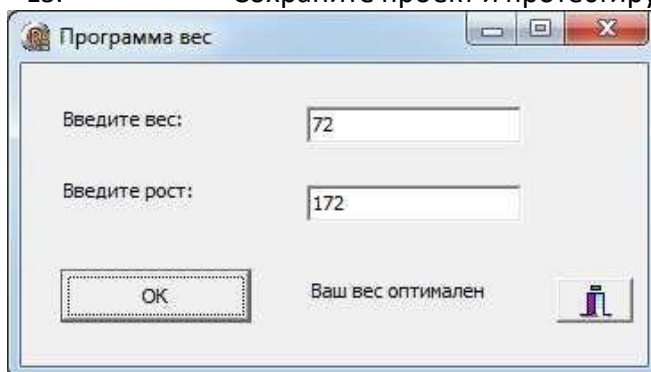
```

procedure TForm1.Button1Click(Sender: TObject);
begin
    faktW:= StrToFloat(Edit1.text);
    Rost:= StrToFloat(Edit2.Text);
    OptW:=Rost - 100;
    Delta:= abs (faktW - OptW);
    if OptW = faktW then Label3.caption := 'Ваш вес оптимален'
    else
    if OptW > faktW then Label3.caption := 'Вам надо поправиться на'
    +Format ('%f ', [Delta])+' кг.'
    else
    Label3.caption := 'Вам надо похудеть на '+Format ('%f ', [Delta])+' кг.'
end;

```

вместо функции **FloatToStr()**. Применим функцию **Format(' %f ',[переменная]): Format ('%f ', [Delta])**.

11. Сохраните проект и протестируйте работу приложения.
12. Озаглавьте окно проекта **Программа Вес**. Создайте кнопку **Close**.
13. Сохраните проект и протестируйте его работу.



Задание.

Используя кнопочные компоненты **TSpeedButton**, разработать программу – калькулятор, выполняющий простейшие действия.

Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её **Калькулятор**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...** под именем **Calc.bdsproj**.
3. Задайте для формы следующие свойства: **Caption** – Calc, **Font - Size** – 10.
4. Разместите на форме одно поле ввода, дайте ему имя **Disp** и очистите свойство **Text**.
5. Для калькулятора мы будем использовать кнопки **TSpeedButton** (группа **Additional**). Их отличие от кнопок **TButton** и **TBitBtn** состоит в том, что они не получают фокуса ввода клавиатуры. Это значит, что их нельзя сделать активными, переходя от элемента к элементу с помощью клавиши **Tab**, и вообще их можно привести в действие только мышкой. Таким образом, на форме будет всего один компонент, принимающий сигналы с клавиатуры — это поле ввода.
6. Добавьте на форму 12 кнопок **TSpeedButton** для цифр, арифметических действий, знака «равно» и операции «сброс». Установите для всех кнопок размеры 25 на 25

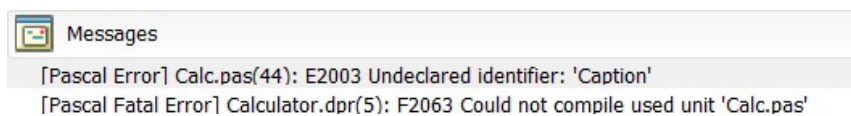


пикселей и разместите их так, как на рисунке.

7. Дайте кнопкам-действиям имена **btnPlus**, **btnMinus**, **btnMul**, **btnDiv** (сложение, вычитание, умножение и деление), а кнопке «равно» — имя **btnCalc**.
8. Для действий и кнопки **C** установите жирный шрифт, а для кнопки **C** дополнительно — красный цвет шрифта.
9. Кнопка **C** должна просто стирать содержимое поля ввода **Disp**. То есть, нужно вызвать метод **Disp.Clear**. Добавьте обработчик события **OnClick** для кнопки **C**.
10. Понятно, что когда пользователь щелкнул по кнопке-цифре, нужно добавить эту цифру в конец текста поля ввода. Конечно, можно для каждой кнопки написать обработчик типа **Disp.Text := Disp.Text + '1'**; Однако можно сделать и более грамотно, определив для всех кнопок общий обработчик. Заметим, что цифра, которую нужно добавить, это и есть текст кнопки,

ее свойство **Caption**. Таким образом, когда мы назначим всем кнопкам один единственный обработчик, в нем нужно как-то определить, какую именно кнопку мы нажали. А такая возможность есть, ведь обработчик имеет параметр **Sender** — это адрес компонента, от которого пришло сообщение. Выделите все кнопки-цифры и определите для них общий обработчик события **onClick: Disp.Text := Disp.Text + Sender.Caption;**

11. Запустите программу. Скорее всего, вы получите сообщение об ошибке:



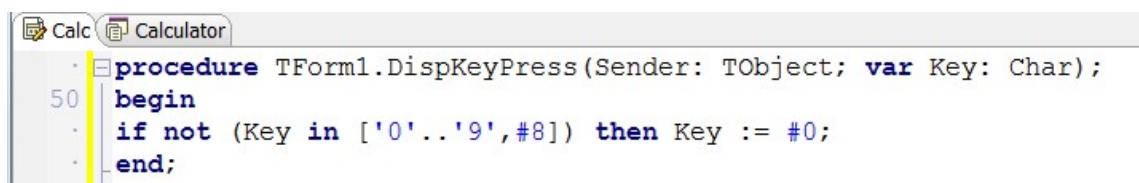
Фраза **Undeclared identifier** означает **Необъявленное имя**, то есть, по мнению **Delphi** объект **Sender** не имеет свойства **Caption**. Обратим внимание на то, что параметр **Sender** объявлен в заголовке процедуры как переменная типа **TObject**, а объект **TObject** действительно не имеет свойства **Caption** (это можно проверить по справочной системе). Но мы знаем, что в нашей программе все компоненты, которые могут вызвать этот обработчик, относятся к типу **TSpeedButton**, у которого свойство **Caption** есть. Поэтому можно сказать, что программа должна воспринимать параметр **Sender** не как просто объект, а как **TSpeedButton: Disp.Text := Disp.Text + TSpeedButton(Sender).Caption;**

12. Исправьте код и запустите программу. Проверьте, как она работает при вводе текста

и букв с клавиатуры и с помощью мыши. Что вам не понравилось?

13. При работе программы вы увидели, что с клавиатуры можно ввести буквы, которые нам совсем не нужны. Когда пользователь нажмет клавишу в поле ввода, возникает событие **OnKeyPress**, которое можно перехватить, установив соответствующий обработчик. В нем все

«ненужные» символы заменяются символом с кодом 0, который не изменяет



содержимое поля. Создайте обработчик события **OnKeyPress** для поля ввода **Disp:**

Разберемся, как работает данная процедура. Обратите внимание, что второй параметр обработчика объявлен как **var Key**, то есть, его можно менять в процедуре. В квадратных скобках записывают множество значений, причем '0'..'9' означает интервал: все символы от '0' до '9'. Символ с кодом 8, обозначаемый как **#8** — это возврат каретки (**Backspace**, удаление символа слева от курсора). Если **Key** — не цифра и не возврат каретки, в эту переменную записывается 0 — нажатие клавиши игнорируется.

Запустите программу и попробуйте вводить буквы.

14. При работе программы вы могли заметить, что при щелчке по кнопке курсор сразу перемещается в начало числа, а хотелось бы, чтобы он оставался справа. Мы сделаем так, чтобы при вводе очередной цифры с помощью кнопки курсор автоматически переводился в конец числа. У компонента **TEdit** есть свойства **SelStart** и **SelLength**, которые определяют начало и длину выделенной части (она обычно обозначается синим фоном). Если **SelLength=0** (ничего не выделено), тогда **SelStart** указывает на позицию курсора в поле ввода. Если установить это свойство равным длине текста, курсор встанет сразу за последним символом.

15. Добавьте в конец обработчика события **OnClick** код: **Disp.SelStart := Length(Disp.Text);** Проверьте работу программы.

16. Далее необходимо организовать вычисления. Нам нужны две переменных для хранения чисел и одна символьная переменная, в которую будем записывать тип операции. Поскольку при расчетах могут получиться числа с дробной частью, для хранения чисел будем использовать вещественные переменные. В простейшем случае для хранения операции можно использовать переменную типа **Char** (один символ), но мы объявим ее как символьную строку, так как при доработке программы могут понадобиться и многосимвольные названия операций. Объявите в начале программы две вещественных переменные **x1** и **x2** типа **Double** и одну символьную строку **oper**.

17. Когда мы нажимаем на одну из кнопок-операций, нужно запомнить введенное число в переменной **x1** и тип операции в переменной **oper**. Тип операции (надпись на кнопке) легко узнать, обратившись к свойству **Caption**. Поэтому можно установить для всех кнопок-операций один обработчик. Выделите все кнопки-операции и создайте для них один обработчик события **OnClick**:

```
x1 := StrToFloat(Disp.Text);  
oper := TSpeedButton(Sender).Caption;
```

18. При нажатии на кнопку «**Равно**» нужно прочитать из поля ввода второе число и выполнить операцию. При этом первое число и тип операции уже должны находиться в переменных **x1** и **oper**. Поскольку в результате деления может получиться число с дробной частью, переменная для хранения результата (назовем ее **res**) тоже должна быть вещественной.

Введите обработчик события **OnClick** для кнопки «**Равно**»:

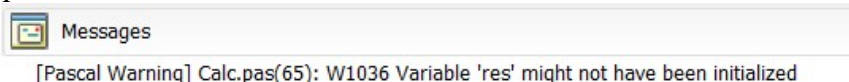
```
procedure TForm1.btnCalcClick(Sender: TObject);  
var res: double;  
begin  
    x2 := StrToFloat(Disp.Text);  
    if oper = '+' then res := x1 + x2;  
    if oper = '-' then res := x1 - x2;  
    if oper = '*' then res := x1 * x2;  
    if oper = '/' then res := x1 / x2;  
    Disp.Text := FloatToStr(res);  
end;
```

19. Запустите программу и проверьте ее работу. Учтите, что для ввода второго числа нужно сначала очистить экран кнопкой «**С**» (позже мы исправим это неудобство).

20. Закройте окно программы и затем запустите ее заново. Введите какое-нибудь число, а затем сразу щелкните по кнопке «**равно**». Что получилось? Попробуйте объяснить этот эффект, учитывая, что при создании глобальной символьной переменной **oper** в нее записана пустая строка.

21. Наверняка вы догадались, что проблема вызвана тем, что мы еще не задали

операцию, а уже попытались что-то вычислить. При этом ни один условный оператор не сработал, и значение переменной **res** не изменялось. А что же в ней было? Откуда взялось это число? Переменная **res** объявлена в процедуре, то есть, она локальная. Память для локальных переменных выделяется в стеке при каждом новом вызове процедуры, причем эти ячейки не обнуляются. Поэтому в переменной **res** в нашем последнем эксперименте остался «мусор» — постороннее значение, которое программа и вывела на экран. Если вы были очень внимательны, можно было заметить, что при трансляции программы в окне **Message** (в нижнем левом углу) было выдано предупреждение:



Это значит «Переменной **res**, возможно, не будет присвоено никакого значения». Формально это не ошибка, и программа может запуститься, однако к предупреждениям нужно относиться внимательно, потому что они могут указать на скрытые логические ошибки (ошибки в алгоритме), как в нашем случае. Какой же выход? Проще всего сделать так: если в переменной **oper** записана пустая строка, мы просто выйдем из процедуры с помощью оператора **Exit**.

22. Добавьте в начало обработчика события **OnClick** для кнопки «Равно»
- ```
if oper = '' then Exit;
```

строку:

Проверьте работу программы. Что вам не нравится?

23. Конечно, очень неудобно, что перед вводом второго числа нужно очищать поле ввода, нажимая на кнопку «С». Хотелось бы делать это автоматически.

24. Запустите стандартную программу **Калькулятор** и посмотрите, в какой момент стирается первое число. Наверное, вы увидели, что число из поля ввода автоматически стирается, когда после нажатия на кнопки-действия или кнопку «Равно» пользователь набирает новое число. Мы введем логическую переменную **newNumber**, которой будем присваивать значение **True** в том случае, если нужно начинать вводить новое число. Объявите глобальную логическую переменную **newNumber**.

25. В конце обработчиков события **OnClick** для кнопок-действий и кнопки «равно» добавьте строку:

```
newNumber := True;
```

26. Очистку экрана будем делать перед вводом нового числа. Дополните обработчики кнопок-цифр. В начало обработчика события **OnClick** для кнопок-цифр добавьте код:

```
if newNumber then begin
 Disp.Clear;
 newNumber := False;
end;
```

27. Учтем в нашей программе, что пользователь может набирать число на клавиатуре. Для этого нужно дополнить аналогичным кодом обработчик события **OnKeyPress** для поля ввода. Измените обработчик события **OnKeyPress** так, как показано ниже.

28. Запустите программу и проверьте ее. Можно ли ввести знаки сложения, вычитания, умножения и деления с клавиатуры? Выполняется ли действие при нажатии на клавишу **Enter**?

```

if not (Key in ['0'..'9',#8]) then Key := #0
else
 if newNumber then begin
 Disp.Clear;
 newNumber := False;
 end;

```

29. При работе программы вы увидели, что пока программа позволяет вводить с клавиатуры только цифры, но не коды действий. Чтобы исправить этот недостаток, мы дополним обработчик **OnKeyPress**.

30. Что нужно сделать, когда нажали символ + на клавиатуре? Проще всего имитировать щелчок по кнопке, то есть, вызвать процедуру-обработчик события **OnClick** для кнопок-действий. Предположим, что этот обработчик называется **btnDivClick** (у вас он может называться по-другому). Но этой функции нужно передать параметр — адрес объекта, пославшего сообщение. Этот объект — кнопка **btnPlus**, поэтому при нажатии клавиши + нужно выполнить команду **btnDivClick ( btnPlus )**;

Обработка нажатия других клавиш строится аналогично. При нажатии клавиши **Enter**, имеющей код 13 (он обозначается в программе как #13), нужно вызвать обработчик события **OnClick** для кнопки

«равно» (она должна называться **btnCalc**).

31. Измените код обработчика **OnKeyPress** следующим образом:

```

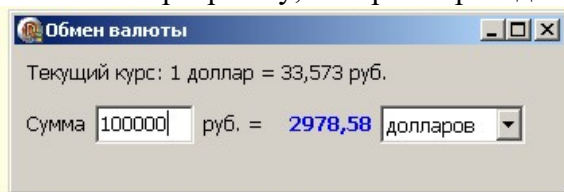
procedure TForm1.DispKeyPress(Sender: TObject; var Key: Char);
begin
 if not (Key in ['0'..'9',#8]) then begin
 case Key of
 '+': btnMinusClick(btnPlus);
 '-': btnMinusClick(btnMinus);
 '*': btnMinusClick(btnMul);
 '/': btnMinusClick(btnDiv);
 #13: btnCalcClick(btnCalc);
 end;
 Key := #0;
 end
 if newNumber then begin
 Disp.Clear;
 newNumber := False;
 end;
 Disp.SelStart := Length(Disp.Text);
 newNumber := True;
end;

```

32. Запустите программу и проверьте ее работу. После этого закройте проект.

Контрольное задание:

Составьте программу, которая переводит суммы из рублей в доллары и евро.



Задание 6.

Разработать программу, которая при нажатии на кнопку «**Output**» выводит сообщение «**Моя первая программа на языке Delphi**», а затем при повторном нажатии на эту же кнопку сообщение исчезает. При повторном выводе размер надписи должен увеличиваться.

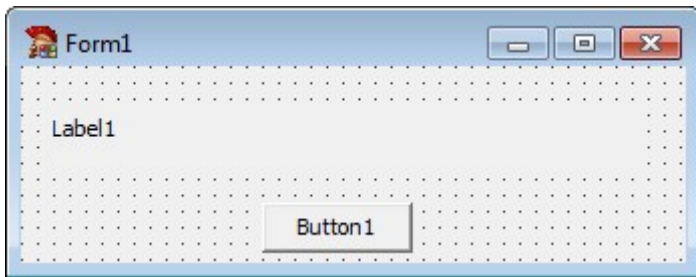
Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте

новый проект.

2. Создайте в папке своей группы новую папку и назовите её **Label**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**

3. Разместите на форме компонент **Label** и кнопку **Button** вкладки палитры компонент-тов **Standard**.



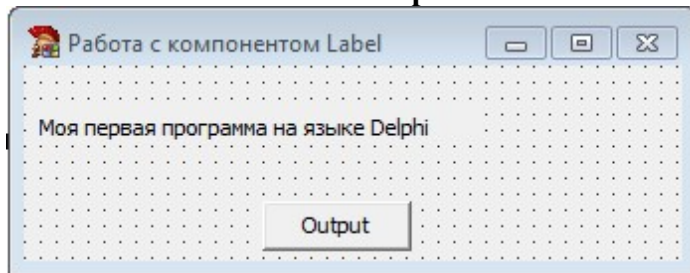
Компоненты ввода — вывода данных можно условно разделить на несколько различных бло-ков: компоненты вывода текстовой информации на экран; однострочные поля ввода текстовой и числовой информации; многострочные поля ввода.

Для вывода определенной информации на экран, кроме уже ранее используемого компонента **Label**, есть и другие компоненты. Текст, который будет отображен, можно задавать как на этапе разработки формы, так и в процессе выполнения программы, присвоив значение свой-ству **Caption**.

4. Задайте для формы заголовок «Работа с компонентом Label».

5. Выделите надпись **Label1**, найдите в **Object Inspector** свойство **Caption** и вставьте новое название надписи **Моя первая программа на языке Delphi**.

6. Выделите кнопку **Button1**, найдите в **Object Inspector** свойство **Caption** и вставьте новое название кнопки **Output**.



7. Перейдите в **Object Inspector** на страницу **Events**, найдите событие **OnClick** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 Label1.Visible:=not Label1.Visible;
 if Label1.Visible then
 Label1.Font.Size:=Label1.Font.Size+1;
end;
```

кнопки **Button1**, напишите следующий программный код:

В этой программе при каждом очередном нажатии происходит изменение свойства **Visible**, вследствие чего надпись то появляется, то исчезает с экрана, а также происходит увеличение свойства **Size**.

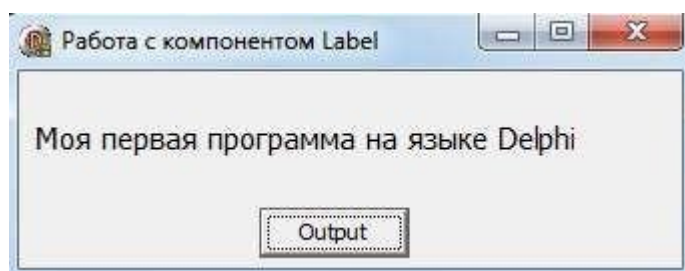
Для вывода определенной информации на экран, кроме уже описанного компонента **Label**, может быть также использован компонент **Panel** с той же самой вкладки или **StaticText** со страницы **Additional**. Они имеют незначительные отличия от компонента **Label**.

Остальные компоненты позволяют вводить и редактировать информацию, включая возможность выделения, копирования, удаления и вставки фрагментов текста. Отметим общие для всех редакторов методы: **Clear** — удалить весь текст, помещенный в редакторе; **ClearSelect**

— удалить выделенный фрагмент текста; **CopyToClipboard** — копировать в буфер выделенный фрагмент, **CutToClipboard** — удалить из текста выделенный фрагмент и поместить его в буфер, **PasteFromClipboard** — копировать текст из буфера в то место редактора, где в данный момент находится курсор.

Для всех редакторов определено дополнительное событие **OnChange**, возникающее, когда изменяется текст, находящийся в редакторе.

Для ввода или вывода одной строки могут использоваться компоненты **Edit** со страницы **Standard** и **MaskEdit** со страницы **Additional**. Основное свойство данных компонентов — это строка, которая либо вводится, либо выводится. Данное свойство имеет имя **Text** и тип **String**, доступное как во время подготовки, так и время выполнения. Логическое свойство **ReadOnly** позволяет запретить изменения, а целочисленное свойство **GetTextLen** выдает текущую длину строки.



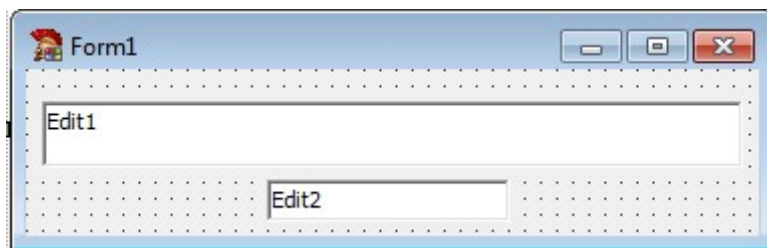
8. Сохраните изменения и запустите проект. Протестируйте его работу.

## Задание 7.

Разработать программу, которая при вводе текста в первый компонент **Edit1**, во втором компоненте **Edit2** отображает реальную длину вводимой строки. Кроме этого, при выходе из компонента **Edit1** его содержимое копируется в буфер обмена и удаляется, а при возвращении в программу появляется снова.

Методические указания по выполнению задания:

1. Создайте в папке своей группы новую папку и назовите её **Edit1**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
2. Разместите на форме два компонента **Edit** вкладки палитры компонентов **Standard**.



3. Задайте для формы заголовок «**Работа с компонентом Edit**».
4. Выделите текстовое поле **Edit1**, найдите в **Object Inspector** свойство **Text** и оставьте его пустым. Аналогичные действия выполните со вторым текстовым полем.

5. Выделите компонент **Edit1**, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnChange** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре текстового поля **Edit1**, напишите следующий программный код:

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
 Edit2.Text:= IntToStr(Edit1.GetTextLen);
end;
```

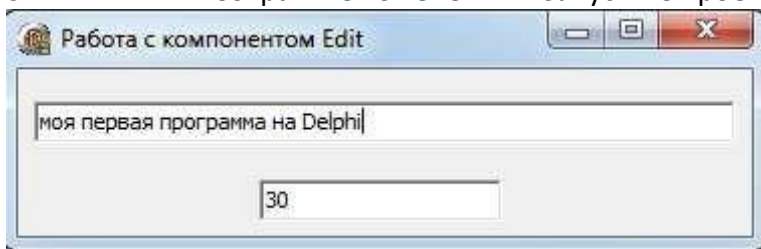
6. Выделите компонент **Edit1**, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnEnter** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре текстового поля **Edit1**, напишите следующий программный код:

```
procedure TForm1.Edit1Enter(Sender: TObject);
begin
 Edit1.PasteFromClipboard;
end;
```

7. Выделите компонент **Edit1**, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnExit** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, но теперь в процедуре текстового поля **Edit1**, напишите следующий программный код:

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
 Edit1.SelectAll;
 Edit1.CopyToClipboard;
 Edit1.Clear;
end;
```

8. Сохраните изменения и запустите проект. Протестируйте его работу.

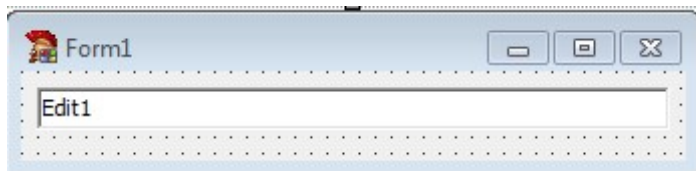


## Задание 8.

Разработать программу, которая запрещает ввод в компонент **Edit1** подряд двух одинаковых символов.

### Методические указания по выполнению задания:

1. Создайте в папке своей группы новую папку и назовите её **Edit2**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**



2. Разместите на форме компонент **Edit** вкладки палитры компонентов **Standard**.

3. Перейдите в код программы (на клавиатуре нажмите клавишу **F12**). Введите в раздел **VAR** глобальную переменную **ch** типа **char**, в которой будет храниться последний нажатый символ.

```
var
 Form1: TForm1;
 ch: char;
```

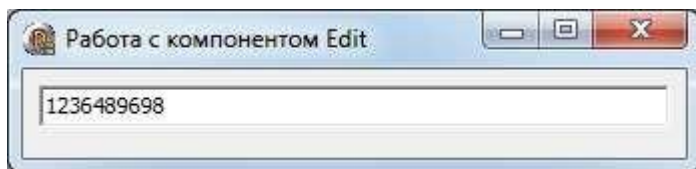
4. Задайте для формы заголовок «**Работа с компонентом Edit**».

5. Выделите текстовое поле **Edit1**, найдите в **Object Inspector** свойство **Text** и оставьте его пустым.

6. Создайте процедуру обработки события **KeyPress** текстового поля **Edit1**, параметр **Key** данной процедуры содержит символ нажатой клавиши. Если вновь введенный символ совпадает с только что нажатым символом, то он игнорируется. В противном случае, новый символ запоминается в переменной **ch**.

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 if ch=key then key:=#0
 else ch:=key;
end;
```

7. Сохраните изменения и запустите проект. Протестируйте его работу.



## Задание 9.

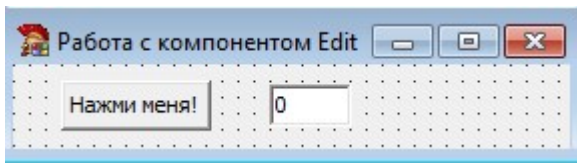
Разработать программу, которая считает количество нажатий на кнопку и выдает это значение в компоненте **Edit**.

### Методические указания по выполнению задания:

1. Создайте в папке своей группы новую папку и назовите её **Edit3**. Сохраните

проект в созданную ранее папку, выполнив команду **File - Save Project as...**

2. Разместите на форме компонент **Edit** и кнопку **Button** вкладки палитры компонентов **Standard**.
3. Используя **Object Inspector**, задайте значения свойств компонентов формы в



соответствии с рисунком.

4. Если в целочисленной переменной **i** будем считать количество нажатий, то процедура обработки события **OnClick** кнопки может быть записана в виде:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 i:=i+1;
 Edit1.Text:=IntToStr(i);
end;
```

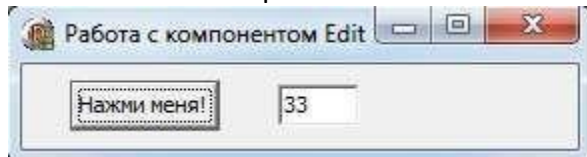
Однако остается вопрос, где описывать данную переменную **i**. Если сделать это внутри данной процедуры, то также необходимо осуществлять обнуление переменной, а это приведет к получению одного и того же результата, равного единице. Следовательно, переменная **i** должна быть глобальной переменной в модуле, а ее начальная инициализация должна происходить в процедуре, которая выполняется всего один раз, и всего один раз происходит это событие. Таким событием является создание формы **OnCreate**, данное событие произойдет один раз и процедура **FormCreate(Sender:TObject)** будет вызвана всего один раз.

5. Перейдите в код программы (на клавиатуре нажмите клавишу **F12**). Введите в раздел **VAR** глобальную переменную **i** типа **integer**, в которой будет храниться последний нажатый символ.

6. Выделите форму, в **Object Inspector** перейдите на вкладку **Events** и найдите событие **OnCreate** и справа от него дважды щелкните мышкой. Оказавшись в коде программы, теперь в процедуре формы **Form1**, напишите следующий программный код:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 i:=0;
end;
```

8. Сохраните изменения и запустите проект. Протестируйте его работу.



9. При возникновении необходимости сделать данную переменную **i** общедоступной, можно поместить описание переменной в интерфейсной части модуля после служебного слова **public**. Именно так, как правило, и поступают. В модуле необходима всего одна переменная

— форма, а все остальные описываются в виде полей. В этом случае описание формы будет иметь вид:

```
type
 TForm1 = class(TForm)
 Edit1: TEdit;
 Button1: TButton;
 procedure Button1Click(Sender: TObject);
 procedure FormCreate(Sender: TObject);
 private
 { Private declarations }
 public
 { Public declarations }
 i: integer;
 end;
var
 Form1: TForm1;
```

В программе для обращения к переменной **i** необходимо писать ее полное имя **Form1.i**. Однако код процедур обработки событий можно и не переписывать, поскольку процедуры обработки

описаны непосредственно в формы, следовательно, данное числовое поле доступно непосредственно.

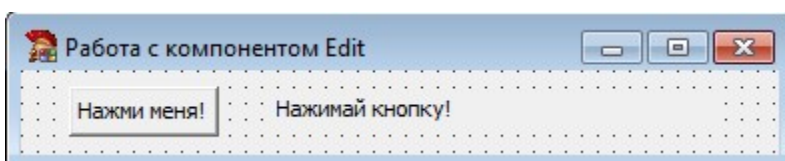
10. Внесите изменения в программный код в соответствии с рисунком. Сохраните проект и протестируйте работу приложения.

11. Данную программу можно легко модифицировать так, чтобы после определенного количества нажатий появлялось некоторое сообщение или кнопка блокировалась, или приложение автоматически закрывалось. Результат можно отображать не только посредством компонента **Edit**, но и через не редактируемый текст, т. е. компонент **Label**, что в данном случае является более естественным.

Свойству **Visible** компонента **Label** присваиваем **False**, т. е. при открытии формы надпись отразаться не будет. Затем, как и ранее, при нажатии на кнопку переменная **i** увеличивается на

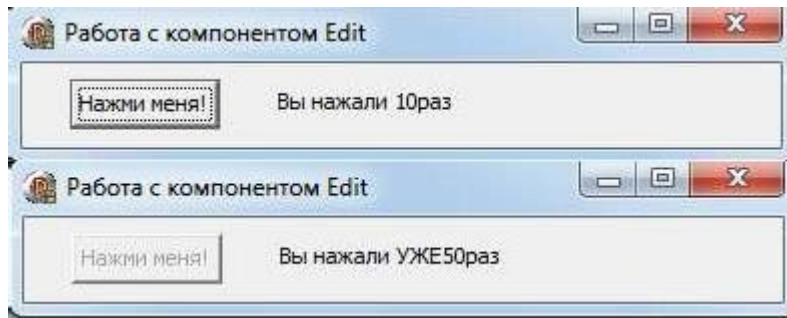
1. Когда значение переменной **i** будет равно **10, 20, 30** или **40** компонент **Label** становится видимым, а свойству **Caption** надписи присваиваем значение «**Вы нажали i раз**». При следующем нажатии надпись становится невидима. Когда **i** станет равной **50**, кнопку необходимо сделать неактивной, для чего необходимо изменить значение свойства **Enabled** с **True** — включено на **False** — выключено.

12. Разместите на форме компонент **Label**. Задайте свойство **Caption** равным



«**Нажимай кнопку!**». Удалите с формы компонент **Edit1**.

13. Перейдите в окно редактирования кода и внесите изменения в код процедуры обработки события **OnClick** кнопки.



### Задание 10.

Разработать программу, которая считывает строку под определенным номером и помещает её в текстовое поле.

#### Методические указания по выполнению задания:

1. Создайте в папке своей группы новую папку и назовите её **Memo1**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
2. Для ввода или вывода нескольких строк могут использоваться компоненты **Memo** со страницы **Standard** и **RichEdit** со страницы **Win32** (полный текстовый редактор для **RTF**-файлов). Многие свойства у данных компонентов аналогичны свойствам компонента **Edit**, однако для возможности доступа к строкам вместо свойства **Text** имеется свойство **Lines**, при выборе которого во время проектирования задается начальное значение строк.

Для доступа к строкам во время выполнения программы также используется свойство **Lines** класса **TString**. Подробнее остановимся на классе **TString**, с которым в последствии мы будем еще встречаться. А именно, этот класс обладает свойствами: **Count** — целочисленное свойство, определяющее количество элементов в списке (в данном случае это будет количество строк в компоненте **Memo**), **Text** — свойство, содержащее все строки списка, **String[Index:Integer]** — свойство, определяющее строку с номером **Index**. Учитывая, что нумерация строк начинается с 0 и свойство **String** является свойством по умолчанию, можно утверждать, что свойства **Memo1.Lines.String[3]** и **Memo1.Lines[3]** эквивалентны и указывают на четвертую строку в компоненте **Memo1**.

Класс **TString** обладает также рядом методов, среди которых отметим следующие: **Add(St:String):integer** добавляет строку **St** и возвращает номер этой строки; **Delete(Index:Integer)** удаляет строку с номером **Index**; **Insert (Index:Integer, St:String)** вставляет строку с номером **Index**, **Clear** полностью уничтожает все содержимое компонента.

Все содержимое компонента можно записать в файл с помощью метода **SaveToFile** или прочесть из файла посредством методом **LoadFromFile**.

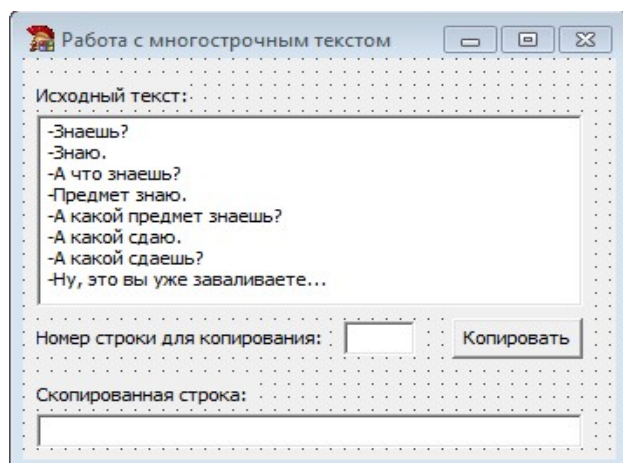
Аналогичным образом можно поступить и с потоком, направив в него весь файл, или прочитать файл из потока.

Важным свойством компонентов **Memo** и **RichEdit** является **ScrollBar**, которое определяет, будет ли окно содержать горизонтальные или вертикальные линейки прокрутки.

Компонент **RichEdit** обладает всеми характеристиками, присущими компоненту **Memo**, од-нако имеет богатые возможности для работы с текстовым форматом **RTF**. Данный формат предполагает возможность разбивать текст на параграфы. Для этого существуют специальные свойства: **SelAttributes** определяет атрибуты выделенного фрагмента и **Paragraph** — атрибуты абзаца.

3. Разместите на форме компоненты **Edit**, компонент **Memo**, компонент **Label**, кнопку **Button** вкладки палитры компонентов **Standard**.

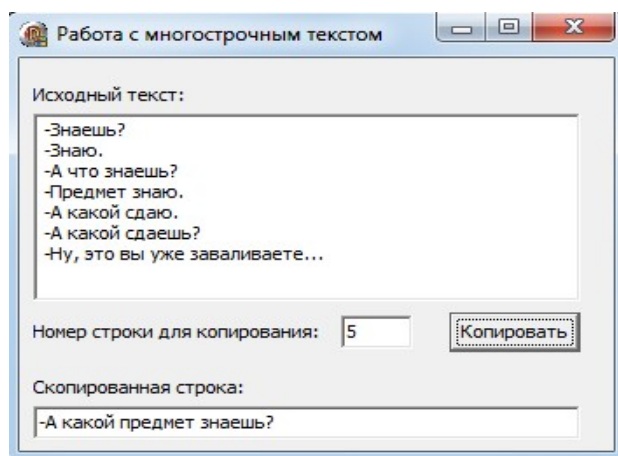
4. Используя **Object Inspector**, задайте значения свойств компонентов формы в соответствии с рисунком.



5. Для реализации решения задачи процедура обработки события **OnClick** кнопки мо-жет быть записана ввиде:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 Edit2.Text:=memo1.Lines[strToInt(edit1.Text)-1];
 Memo1.Lines.Delete(strToInt(edit1.Text)-1);
end;
```

6. Сохраните проект и протестируйте работу приложения.

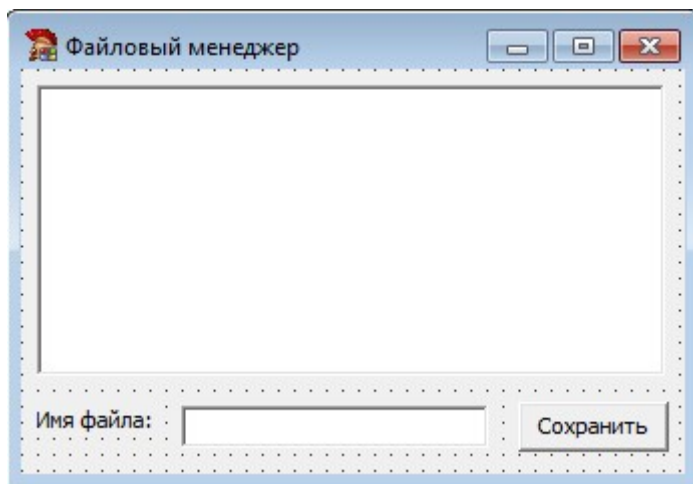


## Задание 11.

Разработать программу, которая сохраняет текст, набранный в поле **Мемо** в файл, имя которого задано в текстовом поле **Edit**.

### Методические указания по выполнению задания:

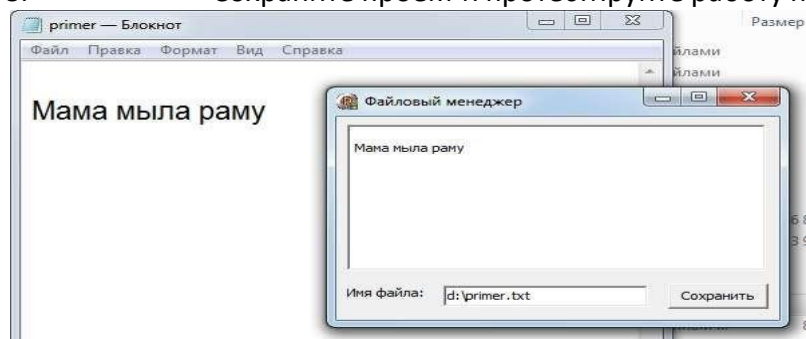
1. Создайте в папке своей группы новую папку и назовите её **Memo2**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
2. Разместите на форме компонент **Edit**, компонент **Memo**, компонент **Label**, кнопку **Button** вкладки палитры компонентов **Standard**.
3. Используя **Object Inspector**, задайте значения свойств компонентов формы в соответствии с рисунком.



4. Процедура обработки события **OnClick** кнопки **Button** будет состоять из одной строки, соответственно, листинг будет иметь следующий вид:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 Memo1.Lines.SaveToFile(Edit1.Text);
end;
```

5. Сохраните проект и протестируйте работу приложения.



Создание проекта с использованием компонентов ввода и отображения чисел, дат и времени.

**Цель работы:** сформировать навыки разработки приложений с использованием компонент ввода и отображения чисел, дат и времени в среде визуального программирования Delphi, изучить особенности их использования

**Оборудование, технические и программные средства:** персональный компьютер, интегрированная среда разработчика **Turbo Delphi**.

Задание 1.

Разработать программу, демонстрирующую действие процедур и функций, оперирующих с системными значениями даты и времени.

Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её **Date1**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
3. Задайте для формы следующие свойства:

| Свойство     | Значение       |
|--------------|----------------|
| BorderStyle  | bsNone         |
| ClientHeight | 441            |
| ClientWidth  | 423            |
| Color        | clCream        |
| Left         | 102            |
| Position     | poScreenCenter |
| Top          | 98             |

4. Разместить на форме компоненты в соответствии с рисунком 2.

5. Задайте для элементов управления **Label** значение свойства

**Caption** в соответствии с ри-сунком. Дляэтого выделите элемент управления и перейдите в **Object Inspector** на страницу **Properties** (свойства).

Button1

Мемо1

часы: нет минуты: нет секунды: нет миллисекунды: нет

Мемо2

год: нет месяц: нет день: нет

Мемо3

дата: нет

Мемо4

время: нет

Мемо5

номер дня недели: нет

6. Задайте для элементов управления **Мемо** значение свойства **Lines** в соответствии с табли-цей. Для этоговыделите элемент управления и перейдите в **Object Inspector** на страницу **Properties** (свойства).

|     | Значение                                                                                                                                                                                      |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Мо1 | <b>DecodeTime (Time, hr, min, sec, msec)</b> процедура, преобразующая текущее значение времени в формат целых чисел по соответствующим переменным.                                            |
| Мо2 | <b>DecodeDate (Date, year, mon, day)</b> процедура, преобразующая текущее значе-ние даты в формат целых чисел по соответствующим переменным.                                                  |
| Мо3 | <b>DateToStr (Date)</b> – функция, возвращающая строку с текущим значением даты в формате day.mon.year                                                                                        |
| Мо4 | <b>TimeToStr (Time)</b> – функция, возвращающая строку с текущим значением времени в формате hr:min:sec                                                                                       |
| Мо5 | <b>DayOfWeek (Date)</b> - функция, возвращающая номер дня недели по следу-ющемусоответствию: 1 – воскресенье, 2 – понедельник, 3 – вторник, 4 – среда, 5 – четверг, 6 – пятница, 7 – суббота. |

7. Задайте для элемента управления **Button1** значение свойства **Caption** равным

X.

8. Введите в окне кода процедуру для отображения даты и времени календаря

```

procedure TForm1.FormCreate(Sender: TObject);
var hr,min,sec,msec:word;
 year,mon,day:word;

begin
 DecodeTime(Time, hr, min, sec, msec);
 Label2.Caption:=IntToStr(hr);
 Label4.Caption:=IntToStr(min);
 Label6.Caption:=IntToStr(sec);
 Label8.Caption:=IntToStr(msec);

 DecodeDate(Date, year, mon, day);
 Label10.Caption:=IntToStr(year);
 Label12.Caption:=IntToStr(mon);
 Label14.Caption:=IntToStr(day);

 Label16.Caption:=DateToStr(Date);
 Label18.Caption:=TimeToStr(Time);
 Label20.Caption:=IntToStr(DayOfWeek(Date));

end;

```

компьютера.

9. Запишите для командной кнопки **Button1** процедуру, которая по щелчку на эту кнопку закрывает форму.

X

DecodeTime (Time, hr, min, sec, msec) процедура, преобразующая текущее значение времени в формат целых чисел по соответствующим переменным.

**часы: 14    минуты: 12    секунды: 21    миллисекунды: 649**

DecodeDate (Date, year, mon, day) процедура, преобразующая текущее значение даты в формат целых чисел по соответствующим переменным.

**год: 2013    месяц: 4    день: 22**

DateToStr (Date) - функция, возвращающая строку с текущим значением даты в формате day.mon.year

**дата: 22.04.2013**

TimeToStr (Time) - функция, возвращающая строку с текущим значением времени в формате hr:min:sec

**время: 14:12:21**

DayOfWeek (Date) - функция, возвращающая номер дня недели по следующему соответствию: 1 - воскресенье, 2 - понедельник, 3 - вторник, 4 - среда, 5 - четверг, 6 - пятница, 7 - суббота.

**номер дня недели: 2**

10. Сохраните изменения и запустите проект, убедитесь в его работоспособности.

## Задание 2.

Разработать приложение, с помощью которого пользователь в одностраничном блокноте вы-бирает однуиз представленных закладок. На рабочем поле блокнота высвечивается соответ-ствующая надпись: год, месяц или день календаря компьютера.

Компонент приложения, содержащий несколько страниц, каждая из которых имеет ярлычок

в виде закладки, называется элементом с закладками. Страницы пользователь может выбирать, щелкая по закладкам (корешкам или ярлычкам).

Методические указания по выполнению задания:

1. В **Delphi** на странице **Win32** Палитры компонентов расположены две составляющие, работающие на программирование элементов с закладками: **TabControl** – одностраничный блокнот; и

**PageControl** – многостраничный блокнот. Данные компоненты являются контейнерами и могут содержать в себе другие элементы или группы.

**TabControl** имеет несколько стилей отображения (свойство **Style**):

- **tsTabs** – стандартные закладки объемного вида;
- **tsButtons** – закладки в виде кнопок;
- **tsFlatButtons** – закладки в виде плоских кнопок. Свойство **Tabs** задает число и названия закладок блокнота.

Свойство **TabIndex** указывает текущую закладку в массиве **Tabs**. Программист может использовать данное свойство для переключения на нужную закладку блокнота. Нумерация начинается с 0. Если значение **TabIndex** равно – 1, то ни одна закладка не выбрана.

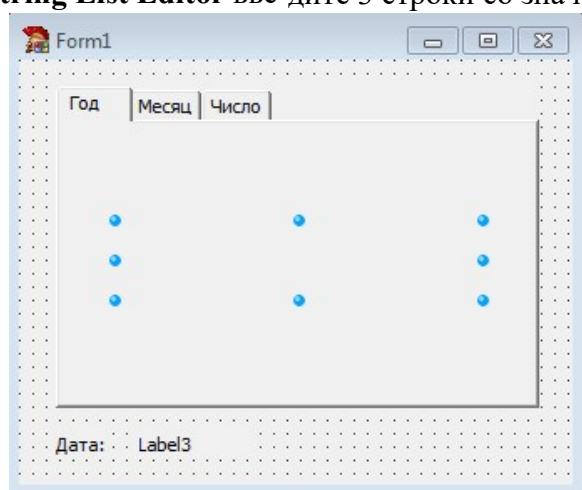
Свойство **HotTrack** задает подсвечивание названия закладки при наведении на неё курсора мыши.

2. Создайте новый проект.  
3. Создайте в папке своей группы новую папку и назовите её **Date2**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**

4. Разместите на форме компоненты **Label**, компонент **TabControl**. В **Label1** будет размещаться надпись блокнота, в **Label2**, **Label3** отображаются текущая дата календаря компьютера.

5. Задайте для элементов управления значение свойства **Caption** в соответствии с рисунком. Для этого выделите элемент управления и перейдите в **Object Inspector** на страницу **Properties** (свойства).

6. Выделите элемент управления **TabControl1**, в **Object Inspector** перейдите на страницу **Properties** (свойства) и найдите свойство **Tabs**, щелкнуть на кнопке . В открывшемся окне **String List Editor** введите 3 строки со значениями год, месяц и число соответственно.



7. Перейдите в окно редактирования кода и опишите в разделе **VAR** переменные **y**, **m**, **d** типа **word** как глобальные переменные модуля для хранения значений даты

```
var
 Form1: TForm1;
 y, m, d: word;
```

календаря компьютера.

```
procedure TForm1.TabControl1Change(Sender: TObject);
begin
case tabcontrol1.TabIndex of
0:
begin
label1.Caption:='';label1.Caption:=IntToStr(y);
end;
1:
begin
label1.Caption:='';
case m of
1: label1.Caption:='январь';
2: label1.Caption:='февраль';
3: label1.Caption:='март';
4: label1.Caption:='апрель';
5: label1.Caption:='май';
6: label1.Caption:='июнь';
7: label1.Caption:='июль';
8: label1.Caption:='август';
9: label1.Caption:='сентябрь';
10: label1.Caption:='октябрь';
11: label1.Caption:='ноябрь';
12: label1.Caption:='декабрь';
end;
end;
2:
begin
label1.Caption:='';
label1.Caption:=IntToStr(d);
end;
end;
end;
```

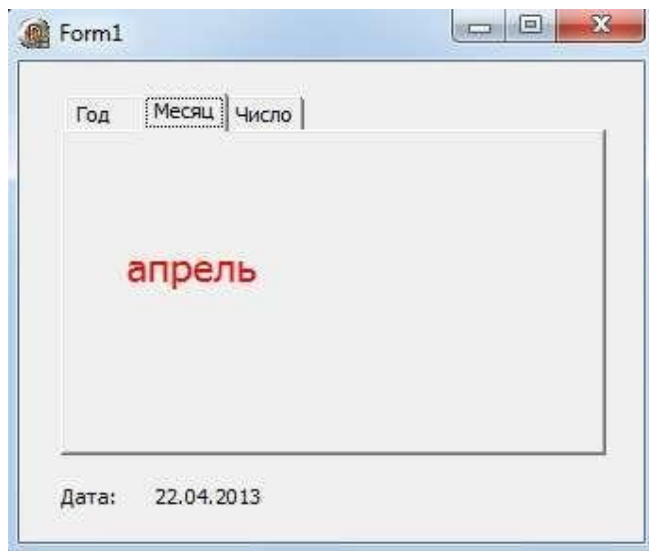
8. Введите в окне редактора кода процедуру для отображения значений даты

```
procedure TForm1.FormCreate(Sender: TObject);
begin
Label3.Caption:= DateToStr(Date); //отображение текущей даты
//календаря компьютера
decodeDate(Date,y,m,d);
Label1.Caption:=intToStr(y); //надпись блокнота
end;
```

календаря компью-тера:

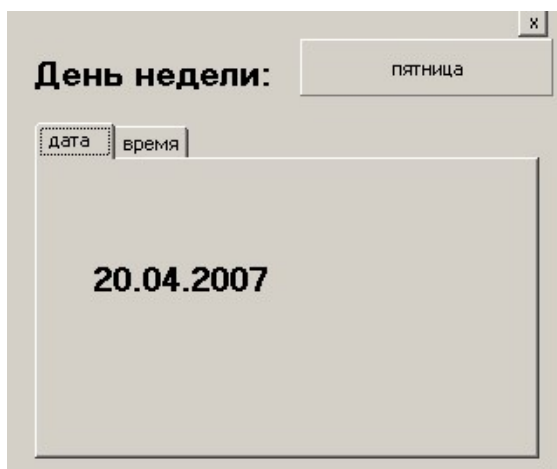
9. Выделите элемент управления **TabControl1**, в **Object Inspector** перейдите на страницу **Events** (события) и найти событие **OnChange**.

10. Введите в окне редактора кода процедуру для работы с элементом управления **TabControl1**.



Контрольное задание 1.

1. Разработайте приложение, с помощью которого пользователь в одностраничном блокноте выбирает одну из представленных вкладок. На рабочем поле блокнота высвечивается соответствующая надпись: дата, время календаря компьютера. На панели отображается день недели.



Создание проекта с использованием кнопочных компонентов

**Цель работы:** сформировать умения использования переключателей при создании проекта Delphi, изучить их основные свойства; сформировать умения по созданию процедур на основе событий компонентов.

**Оборудование, технические и программные средства:** персональный компьютер, интегрированная среда разработчика **Turbo Delphi**.

### Задание 1.

Разработать программу, с помощью которой пользователь мог бы выполнить следующее. После запуска программы появляется изображение, аналогичное рисунку. Пользователь по своему усмотрению выбирает один переключатель в группе. Каждому переключателю соответствует определенный балл. В зависимости от суммы набранных баллов появляется одно из сообщений «Вы пессимист», «Вы реалист» или «Вы оптимист».

Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её **Radio1**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
3. Разместить на форме компоненты в соответствии с рисунком.



4. Выделите компонент **RadioGroup1**, перейдите в **Object Inspector** на страницу **Properties**, найдите свойство **Caption** и удалите заголовок. Свойству **Columns**, определяющему количество колонок, в которые будут отображаться переключатели, присвойте значение **5**.
5. Вызовите **String List Editor**, дважды щелкнув мышкой рядом со свойством **Items**, введите **5** строк со значениями 1, 2, 3, 4 и 5 соответственно.
6. Аналогичные действия проделайте с остальными компонентами **RadioGroup**. Можно выделить элементы управления с нажатой клавишей **Shift** и задать свойства,

которые будут заданы для всех выделенных элементов управления.

7. Для того чтобы суммировать набираемые пользователем баллы, в процедуру обработки события **RadioGroup1.OnClick** вставьте следующий код:

```
sum:=0;
with RadioGroup1 do
if ItemIndex>=0 then sum:=sum+ItemIndex+1;
```

Единицу необходимо прибавлять, т.к. индекс первого переключателя равен 0, но соответствует 1баллу. Целочисленную переменную **sum** необходимо описать в разделе **var** модуля.

8. Вставьте в процедуру обработки событий **RadioGroup2.OnClick**, **RadioGroup3.OnClick** и **RadioGroup4.OnClick** аналогичные коды, но без обнуления переменной **sum**, т.к. обнуление необходимо лишь один раз перед началом суммирования.

9. Выведем на контрольную панель итоговое сообщение в зависимости от набранной суммы баллов. Для этого в процедуру обработки события **RadioGroup4.OnClick** добавим код:

```
case sum of
4..9: Panel1.Caption:='Вы пессимист';
10..15: Panel1.Caption:='Вы реалист';
16..20: Panel1.Caption:='Вы оптимист';
end;
```

10. Выведем сообщение об окончании тестирования, добавив в процедуру обработки события

**RadioGroup4.OnClick код:** `ShowMessage('Конец теста');`

В результате выполнения этого оператора появится информационное окно со словами «Конец теста» и единственной кнопкой **Ok**.

После нажатия кнопки **Ok** в информационном окне сделайте недоступными все **RadioGroup**, а всепереключатели в них - невыбранными. Для этого свойствам **Enabled** всех **RadioGroup** присвоить значение **False**, а свойствам **ItemIndex** - значение **-1**.

11. Для контроля правильности работы программы выведите на панель набранную пользователем суммубаллов. Для этого заголовку соответствующей панели присвоить значение **IntToStr(sum)**.

12. Запустите программу и убедитесь, что верная сумма баллов получается лишь при последовательном выборе переключателей сначала из **RadioGroup1**, затем из **RadioGroup2** и т.д. Если порядок был нарушен, или пользователь, изменив решение, выбрал другой переключатель в одной и той же группе, то результат будет неверным. Чтобы этого не случилось, сделайте доступной только ту группу переключателей, в которой необходимо сделать выбор. Для этого первоначально свойству **Enabled** всех **RadioGroup**, кроме **RadioGroup1**, присвойте значение **False**. В процедуре обработки события **RadioGroup1Click** добавить код, присваивающий свойству **Enabled RadioGroup2** значение **True** и т.д.

13. Сделайте возможным повторный запуск программы. Для этого разместите на форме кнопку **Button1**, свойству **Caption** которой присвойте значение **Retry**, а в процедуре обработки события **Button1Click** вставьте код, делающий доступной **RadioGroup1**.

14. Озаглавьте окно проекта **Проверь себя**.

15. Сохраните изменения и запустите проект, убедитесь в его работоспособности.

## Задание 2.

Разработать приложение, с помощью которого можно вычислить время падения тела с некоторой высоты при условии, что высота может задаваться в метрах, сантиметрах и дюймах.

### Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её **Radio2**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**
3. Разместить на форме компоненты в соответствии с рисунком.

4. Создадим процедуру обработки события **Click** для кнопки **Вычислить**:

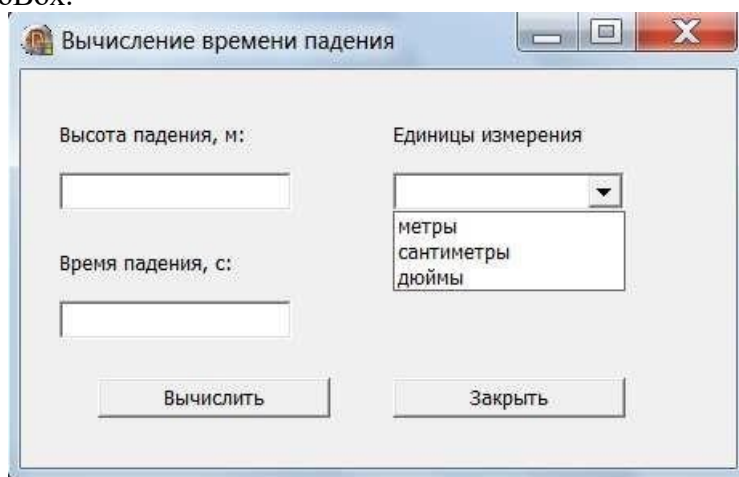
```
procedure TForm1.Button1Click(Sender: TObject);
 Const g=9.81;
 Var h,t : Real;
Begin
 h:=StrToFloat(Edit1.Text);
 Case radioGroup1.ItemIndex of
 0: t:=sqrt(2*h/g); {Высота задается в метрах}
 1: t:=sqrt(2*h/100/g); {Высота задается в санти-метрах}
 2: t:=sqrt(2*h*2.54/100/g); {Высота задается в дюймах}
 End;
 Edit2.Text:=FloatToStr(t);
end;
```

5. Запустите программу и проверьте её работоспособность.

6. Самостоятельно напишите обработчик события кнопки **Заккрыть**.

7. Недостаток компонента **RadioGroup** заключается в том, что имеется возможность определить только номер выбранной альтернативы, а не ее текстовое содержание. Для того чтобы определить и текстовое содержание альтернативы, можно использовать комбинированную строку, компонент **ComboBox**. Ком-бинированная строка ввода объединяет в себе свойство строки и списка. В обычном состоянии она имеет вид строки **Edit** со стоящей рядом кнопкой с изображением направленной вниз стрелки. Если нажать эту кнопку, то появится список строк, где можно выбрать произвольную. Данный компонент имеет свойство **Items**, поэтому задание альтернатив, которые в данном случае будут раскрывающимся списком, происходит аналогичным образом. Однако на этапе выполнения допустимо свойство **Text**, в котором находится выбранная из списка строка. При работе с данным компонентом необходимо различать свойства **ComboBox.Items**, **Text** и **ComboBox.Text**. Если первое — это свойство, где находятся все строки списка, включая разделители (это свойство имеет подобный смысл и для **RadioGroup**), то второе содержит выбранную из списка строку.

8. Измените проект так, чтобы выбор единицы измерения осуществлялся с помощью компонента **ComboBox**.



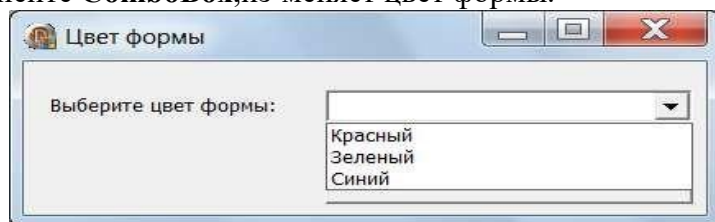
```
procedure TForm1.Button1Click(Sender: TObject);
 Const g=9.81;
 Var h,t : Real;
Begin
 h:=StrToFloat(Edit1.Text);
 if ComboBox1.Text='метры' then t:=sqrt(2*h/g);
 {Высота задается в метрах}
 if ComboBox1.Text='сантиметры' then t:=sqrt(2*h/100/g);
 {Высота задается в сантиметрах}
 if ComboBox1.Text='дюймы' then t:=sqrt(2*h*2.54/100/g);
 {Высота задается в дюймах}
 Edit2.Text:=FloatToStr(t);
end;
```

9. Выбор между компонентами **RadioGroup** и **ComboBox** во многом зависит от вкусов программистов, поскольку они выполняют одинаковую роль, а имеют различные внешний вид и применение. Необходимо отметить, что в языке **Delphi** есть большое количество компонентов, имеющих одинаковую область применения, однако отличающихся некоторыми частными внешними особенностями и доступными для применения методами.

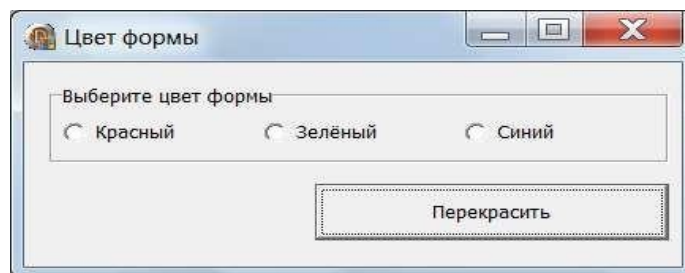
10. Сохраните изменения в проекте.

### Контрольное задание 1.

1. Разработайте приложение, которое при выборе определенного цвета в компоненте **ComboBox**,из-меняет цвет формы.



2. Разработайте приложение, которое при выборе определенного цвета в компоненте **RadioGroup**,из-меняет цвет формы.

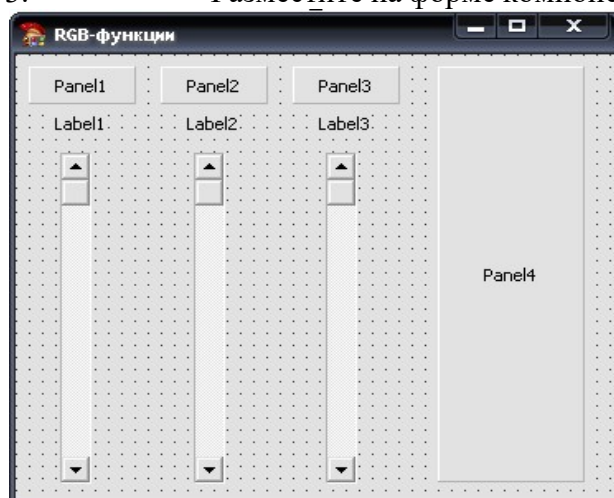


### Задание 3.

Разработать проект демонстрации работы RGB – функций (установок цвета по трем составляю-щим) с помощью полос прокрутки. Каждый бегунок полос прокрутки должен будет менять вклад RGB – компонента, отображающийся на панели как цвет, а на метке как число. Результи-рующий цвет долженотображаться на панели.

Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.
2. Создайте в папке своей группы новую папку и назовите её **ScrollBox1**. Сохраните проект в создан-нуюранее папку, выполнив команду **File - Save Project as...**
3. Разместите на форме компоненты в соответствии с рисунком.



**Полоса прокрутки** – классический элемент оконного интерфейса, использующийся для

скроллинга информации, которая не помещается целиком в область вывода. В объектах просмотра и редактирования **Windows** этот элемент появляется при необходимости автоматически.

В палитре VCL-компонент для создания полос прокрутки как самостоятельных элементов введен отдельный компонент:  **ScrollBar** – самостоятельная полоса прокрутки, может быть горизонтальной или вертикальной, что определяется свойством **Kind=sbHorizontal/sbVertical**.

Свойствами **Min** и **Max** можно задать края диапазона целых значений, соответствующих положению бегунка полосы (свойство **Position**). Обычно также выбирают размеры скачков бегунка при щелчке на стрелках или на свободной полосе, определяя свойства **SmallChange** и **LargeChange**.

При перемещении бегунка **ScrollBar** генерируется событие **OnChange** (при попытке перемещения – **OnScroll**). При этом можно получать и обрабатывать данные как свойство **Position**. Во время работы программы значения свойств **Position**, **Min**, **Max** можно задавать с помощью метода **SetParams**.

4. Задайте для элемента управления **ScrollBar1** значение свойства **Name=RedBar** и установите значение свойств **Max=255**, **Position=122**.

5. Задайте для элемента управления **Scrollbar2** значение свойства **Name=GreenBar** и установите значение свойств **Max=255**, **Position=122**.

6. Задайте для элемента управления **Scrollbar3** значение свойства **Name=BlueBar** и установите значение свойств **Max=255**, **Position=122**.

7. Выделите элемент **RedBar**, в **Object Inspector** перейдите на вкладку **Events**. Найдите событие **OnChange**, справа от него в поле сделайте двойной щелчок левой кнопкой мыши. Оказавшись в коде программы, введите следующий код:

```
procedure TForm1.RedBarChange(Sender: TObject);
begin
 Panel1.Color:=TColorRef(RGB (RedBar.Position,0,0));
 Label1.Caption:=IntToStr (RedBar.Position);
 Panel4.Color:=TColorRef (RGB (RedBar.Position,GreenBar.Position,BlueBar.Position))
end;
```

Функция **TColorRef** преобразует значение цветовых составляющих в число типа **LongInt**.

8. Аналогично запишите процедуры обработки события **OnChange** для элементов **GreenBar** и **BlueBar**.

9. Задайте имя формы проекта **RGB-функции**. Создайте кнопку **Close**.

10. Вызов процедур обработки событий **OnChange** поместите в событие при создании формы **OnCreate** для **Form1**.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 BlueBar.OnChange(Sender);
 RedBar.OnChange(Sender);
 GreenBar.OnChange(Sender);
end;
```

11. Сохраните изменения и запустите проект, убедитесь в его работоспособности.

#### Задание 4.

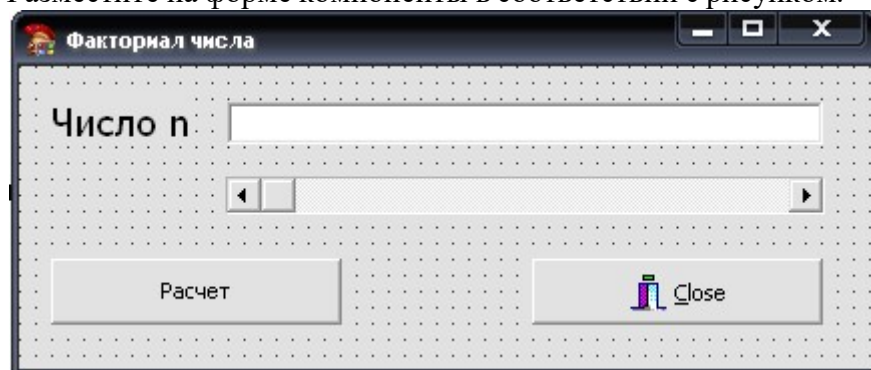
Разработать проект, который позволяет пользователю вычислить факториал числа. Число, для которого рассчитывается факториал, выбирается с помощью вертикальной полосы прокрутки. При щелчке по кнопке «Расчет», меняется надпись «Число n» на «N!» и в строке ввода выводится значение факториала числа.

Методические указания по выполнению задания:

1. Запустите интегрированную среду разработчика **Turbo Delphi**. Создайте новый проект.

2. Создайте в папке своей группы новую папку и назовите её **ScrollBar2**. Сохраните проект в созданную ранее папку, выполнив команду **File - Save Project as...**

3. Разместите на форме компоненты в соответствии с рисунком.



4. Для элемента управления **ScrollBar1** установите значение свойств **Max=10, Position=5**.

5. Выделите элемент **ScrollBar1**, в **Object Inspector** перейдите на вкладку **Events**. Найдите событие **OnChange**, справа от него в поле сделать двойной щелчок левой кнопкой

```
procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
 Edit1.Text:=IntToStr(ScrollBar1.Position);
end;
```

мыши. Оказавшись в коде программы, ввести следующий код:

6. Запишите процедуру обработки события **OnClick** кнопки «Расчет». При расчете факториала числа воспользоваться конструкцией цикла.

7. Задайте имя формы проекта **Факториал числа**. Создайте кнопку **Close**.

8. Сохраните изменения в проекте.

Создание проекта с использованием компонентов стандартных диалогов и системы меню

**Цель работы:** сформировать умения использования компонентов стандартных диалогов и системы меню, изучить основные свойства; сформировать умения по созданию процедур на основе событий компонентов.

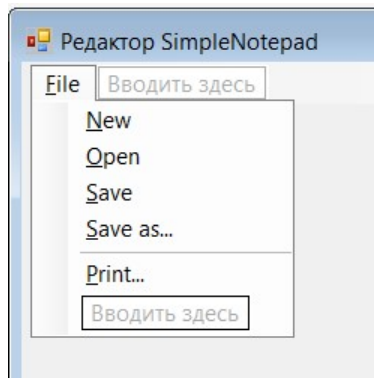
**Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

Задание:

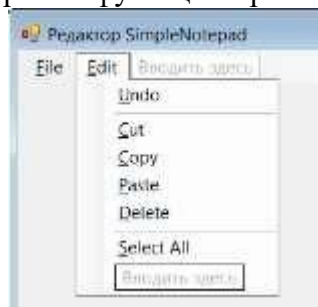
Разработать приложение **SimpleNotepad**, представляющее собой простейший текстовый редактор. Использовать в приложении компонент для работы с меню и стандартные окна диалога.

Методические указания по выполнению задания:

1. Запустите среду программирования **Visual Studio**. Создайте новое **Приложение Windows Forms**. Имя проекта и приложения **SimpleNotepad**.
2. Измените свойство **Name** формы на **SimpleNotepadForm**.
3. Перейдите в окно **Обозреватель решений** и переименуйте файл **Form1.cs** на **SimpleNotepadForm.cs**.
4. Измените размеры окна формы. Измените заголовок окна на **Редактор SimpleNotepad**.
5. Добавьте в окно приложения элемент управления **MenuStrip**.
6. В поле **Вводить здесь** меню введите строку **&File**. В результате в окне приложения появится меню **File**. Обратите внимание, что первая буква в названии подчеркнута. Это получилось по-тому, что перед ней стоит префикс **&**. Этим префиксом отмечается буква, предназначенная для ускоренного выбора меню при помощи клавиатуры.
7. Создайте меню **File** в соответствии с рисунком. Для вставки разделительной черты необходимо вызвать контекстное меню и выбрать команду **Separator**.

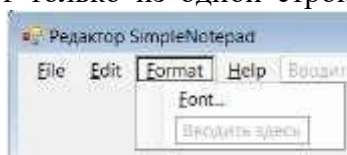


8. Создайте меню **Edit** в соответствии с рисунком. В меню **Edit** реализованы стандартные функции приложения **Блокнот**.

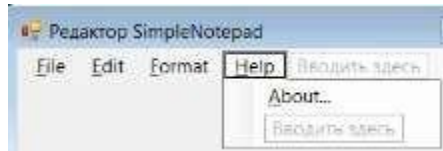


9. Меню **Format** состоит только из одной строки **Font**, с помощью которой

пользователь может изменить шрифт текста.



10. Меню **Help** состоит только из одной строки **About**.



11. Переименуйте идентификаторы меню и строк меню таким образом, чтобы с ними было удобнее работать в программе. Для этого в окне дизайнера формы щелкните правой кнопкой мыши главное меню приложения и затем выберите из контекстного меню строку **Правка DropDownItems**. В окне **Редактор коллекций элементов** отредактируйте имена меню и строк меню, изменив значение свойства **Name** соответствующего элемента. При этом меню верхнего уровня должны называться **menuFile**, **menuEdit**, **menuFormat**, **menuHelp**. Имена строк формируются путем добавления к имени меню текста, отображаемого в строке меню. Например, строка **New** называется **menuFileNew**.

12. Запустите приложение. Убедитесь, что меню корректно отображается, и вы можете выбирать его строки.

13. Редактирование текста в приложении будет выполнять компонент **RichTextBox**. Перетащите компонент **RichTextBox** с панели элементов в окно приложения. Задайте значение свойства **Dock** данного компонента равным **Fill**, для того чтобы он занимал все окно приложения.

14. Создайте обработчики событий, необходимые для выполнения таких функций как создание нового документа, открытие и сохранение документа.

15. Добавьте на форму компонент **OpenFileDialog** с вкладки **Диалоговые окна** панели элементов. Выделите пункт меню **Open...** и дважды щелкните по нему левой кнопкой мыши, для того чтобы добавить обработчик события. В обработчик события добавьте следующий код для отображения диалогового окна открытия файла.

```
private void menuFileOpen_Click(object sender, EventArgs e)
{
 if (openFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK && openFileDialog1.FileName.Length > 0)
 {
 try
 {
 richTextBox1.LoadFile(openFileDialog1.FileName, RichTextBoxStreamType.RichText);
 }
 catch (System.ArgumentException ex)
 {
 richTextBox1.LoadFile(openFileDialog1.FileName, RichTextBoxStreamType.PlainText);
 }
 this.Text = "Файл[" + openFileDialog1.FileName + "]";
 }
}
```

16. Выделите компонент **openFileDialog1**, на панели **Свойства** найдите свойство **Filter** и введите следующую строку **RTF files|\*.rtf|Text files|\*.txt|All files|\*.\***

17. Добавьте на форму компонент **SaveFileDialog** с вкладки **Диалоговые окна** панели элементов. Выделите пункт меню **Save As...** и дважды щелкните по нему левой кнопкой мыши, для того чтобы добавить обработчик события. В обработчик события добавьте следующий код для отображения диалогового окна **Сохранение файла**. Этот код сохраняет текст введенный в элемент управления **richTextBox**, в текстовый файл в указанной папке. Метод **ShowDialog** отображает диалоговое окно, а затем с помощью поля **DialogResult.OK** осуществляется проверка нажата ли кнопка **OK**.

```
private void menuFileSave_Click(object sender, EventArgs e)
{
 if (saveFileDialog1.ShowDialog()==System.Windows.Forms.DialogResult.OK && saveFileDialog1.FileName.Length>0)
 {
 richTextBox1.SaveFile(saveFileDialog1.FileName);
 this.Text = "Файл [" + saveFileDialog1.FileName + "]";
 }
}
```

18. Выделите компонент **saveFileDialog1**, на панели **Свойства** найдите свойство **Filter** и введите следующую строку **RTF files|\*.rtf**, найдите свойство **FileName** и задайте значение **doc1.rtf**

19. Добавьте в окно дизайнера форм компоненты **PrintDocument**, **PrintDialog** вкладки **Печать** панели элементов.



20. Компонент **PrintDocument** предназначен для вывода данных документа на принтер. Свойства компонента **PrintDocument** описывают, как именно нужно распечатывать документ. В свойство **DocumentName** данного компонента запишите строку **SimpleNotepad Document**. Эта строка будет идентифицировать документ при отображении состояния очереди печати. Значения остальных свойств оставьте без изменения.

21. С помощью компонента **PrintDialog** приложение выведет на экран стандартное диалоговое окно печати документа. Задайте для данного компонента значение свойства **Document** равным **printDocument1**. Этим обеспечивается связь компонента **printDialog** с компонентом **PrintDocument**.

22. Для работы с классами, предназначенными для выполнения операций с

```
using System.IO;
using System.Drawing.Printing;
```

потоками и печати, добавьте в начало программы следующие строки:

23. Добавьте в класса **SimpleNotepadForm** поля **m\_myReader** (для печати содержимого редактора текста) и **m\_PrintPageNumber** (номер текущей распечатываемой страницы документа).

```
public partial class SimpleNotepadForm : Form
{
 private StringReader m_myReader;
 private uint m_PrintPageNumber

 public SimpleNotepadForm()
 {
```

24. Создайте обработчик события печати документа.

```
private void menuFilePrint_Click(object sender, EventArgs e)
{
 m_PrintPageNumber = 1;
 string strText = this.richTextBox1.Text;
 m_myReader = new StringReader(strText);
 Margins margins = new Margins(100, 50, 50, 50);
 printDocument1.DefaultPageSettings.Margins = margins;
 if (printDialog1.ShowDialog()==DialogResult.OK)
 {
 this.printDocument1.Print();
 }
 m_myReader.Close();
}
```

Печать документа будет начинаться с первой страницы, поэтому в поле **m\_PrintPageNumber** записывается значение 1. Далее выполняется чтение текущего содержимого окна редактирования текста в поток **m\_myReader** класса **StringReader**. Далее задаются границы отступов на

распечатываемой странице и отображается диалоговое окно печати документа. Если пользователь щелкает в этом окне кнопку **OK**, документ **printDocument1** отправляется на печать методом **Print**. Далее ненужный более поток **m\_myReader** закрывается методом **Close**.

25. На данном этапе приложение еще не в состоянии распечатать документ. Причина этого в том, что приложение пока еще не знает, каким именно образом нужно печатать документ.

26. Чтобы в нашем приложении заработала функция печати, необходимо создать обработчик события **PrintPage**. Для этого нужно дважды щелкнуть левой клавишей мыши значок компонента **printDocument1**. В тело обработчика событий вставьте программный код из текстового файла **Print.txt**. Комментарии в тексте обработчика событий **PrintPage** поясняют назначение отдельных программных строк. Заметим, что для полного понимания действий, выполняемых нашим обработчиком событий, требуется предварительное знакомство с графической подсистемой **Graphics Device Interface Plus (GDI+)**, реализованной компанией **Microsoft** в рамках библиотеки классов **.NET Framework**. Пока же нужно отметить, что приложение распечатывает текст построчно в цикле. После завершения печати всех строк текущей страницы обработчик событий **PrintPage** печатает верхний и нижний колонтитулы, а также рисует горизонтальные линии, отделяющие текст колонтитулов от текста документа.

27. Измените меню **File**, добавив пункт **Exit**, при выборе которого окно редактора текста должно быть закрыто. Добавьте обработчик события для данного пункта меню:

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
 this.Close();
}
```

Однако при закрытии окна приложения возникает проблема: окно редактора текста будет закрыто и в том случае, если пользователь не сохранил сделанные им изменения. Чтобы решить эту проблему, нам нужно каким-то образом отслеживать наличие изменений в окне редактирования текста. Определите в классе **SimpleNotepadForm** поле **m\_DocumentChanged**, в котором будет храниться флаг, отмечающий изменения, сделанные пользователем в документе. В новом или только что загруженном документе изменений нет, поэтому начальное значение этого флага равно **false**.

```
private bool m_DocumentChanged = false;
```

28. Создайте обработчик события **richTextBox1\_TextChanged**, выполнив двойной щелчок по компоненту **richTextBox**. Этот обработчик получит управление, как только пользователь внесет любые изменения в содержимое редактируемого документа.

```
private void richTextBox1_TextChanged(object sender, EventArgs e)
{
 m_DocumentChanged = true;
}
```

29. Если пользователь редактировал документ, а потом решил создать новый, выбрав из меню **File** строку

**New**, изменения, внесенные в старый документ, могут быть потеряны.

30. Чтобы избежать этого, проверьте флаг **m\_DocumentChanged** перед тем как очищать содержимое редактора текста. Если в редакторе есть не сохраненные изменения, необходимо выполнить сохранение документа. Отредактируйте код обработчика события пункта меню **New** следующим образом:

```
private void menuFileNew_Click(object sender, EventArgs e)
{
 if (m_DocumentChanged)
 {
 if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK && saveFileDialog1.FileName.Length > 0)
 {
 richTextBox1.SaveFile(saveFileDialog1.FileName);
 this.Text = "Файл [" + saveFileDialog1.FileName + "]";
 m_DocumentChanged = false;
 }
 }
 richTextBox1.Clear();
}
```

31. Отредактируйте код обработчика события пунктов меню **Save** и **Save As**, добавив строку кода **m\_DocumentChanged=false**

32. Измените обработчик события пункта меню **Exit** следующим образом:

```
private void exitToolStripMenuItem_Click(object sender, EventArgs e)
{
 if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK && saveFileDialog1.FileName.Length > 0)
 {
 richTextBox1.SaveFile(saveFileDialog1.FileName);
 this.Text = "Файл [" + saveFileDialog1.FileName + "]";
 m_DocumentChanged = false;
 }
 this.Close();
}
```

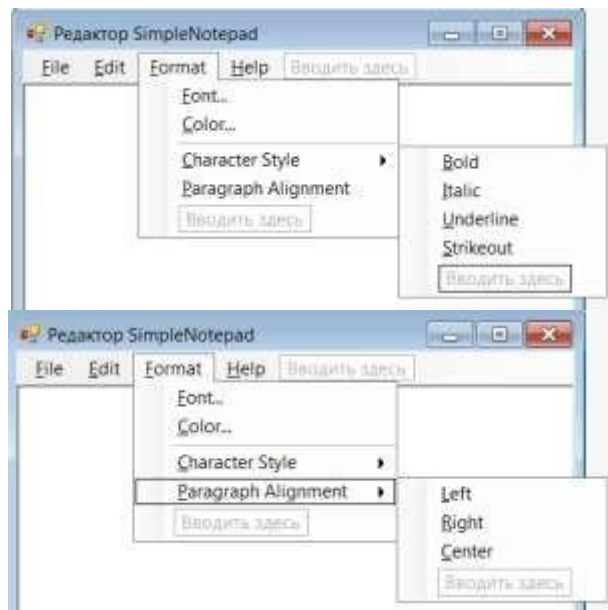
33. Надо выполнить еще одну проверку флага **m\_DocumentChanged** в методе **Dispose**, который вызывается при закрытии окна приложения. Для этого на панели Обозреватель решений выделите **SimpleNotepadForm.Designer.cs**, найдите метод **Dispose** и внесите необходимые изменения.

```
protected override void Dispose(bool disposing)
{
 if (saveFileDialog1.ShowDialog() == System.Windows.Forms.DialogResult.OK && saveFileDialog1.FileName.Length > 0)
 {
 richTextBox1.SaveFile(saveFileDialog1.FileName);
 this.Text = "Файл [" + saveFileDialog1.FileName + "]";
 m_DocumentChanged = false;
 }
 if (disposing && (components != null))
 {
 components.Dispose();
 }
 base.Dispose(disposing);
}
```

Теперь, когда пользователь попытается закрыть программу с помощью соответствующей кнопки заголовка окна, не сохранив сделанные изменения, на экране появится стандартное диалоговое окно, предлагающее ему сохранить документ.

34. Создайте обработчики событий пунктов меню **Edit**. Реализация данных функций является

35. Измените меню **Format**, добавив в него строку **Color**, а также два меню второго уровня – **Character Style** и **Paragraph Alignment**. Измените имена строк меню таким



образом, чтобы с ними было легче работать в программе.

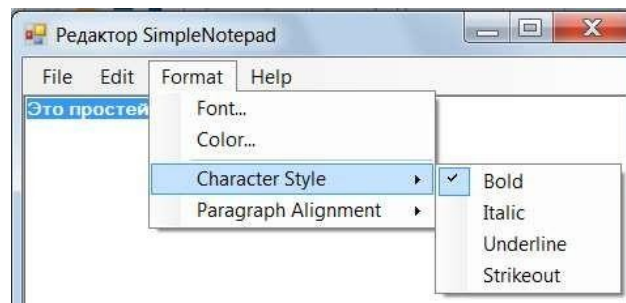
```
private void menuFormatFontCharacterStyleBold_Click(object sender, EventArgs e)
{
 if (richTextBox1.SelectionFont!=null)
 {
 System.Drawing.Font currentFont = richTextBox1.SelectionFont;
 System.Drawing.FontStyle newFontStyle;
 if (richTextBox1.SelectionFont.Bold==true)
 {
 newFontStyle = FontStyle.Regular;
 }
 else
 {
 newFontStyle = FontStyle.Bold;
 }
 richTextBox1.SelectionFont = new Font(currentFont.FontFamily, currentFont.Size, newFontStyle);
 CheckMenuFontCharacterStyle();
 }
}
```

Получив управление данный метод прежде всего, определяет шрифт фрагмента текста, выделенного пользователем, анализируя свойство **richTextBox1.SelectionFont**. Если шрифт определить не удалось, и это свойство содержит значение **null**, программа не делает никаких изменений. В противном случае программа сохраняет текущий шрифт в переменной **currentFont** класса **System.Drawing.Font**. Далее метод проверяет, был ли выделен фрагмент текста жирным шрифтом, анализируя свойство **richTextBox1.SelectionFont.Bold**. Если это свойство содержит значение **true**, то метод снимает выделение, если нет, то устанавливает его. Для снятия выделения программа записывает в переменную **newFontStyle** значение **FontStyle.Regular**, а для установки — значение **FontStyle.Bold**.

36. При выделении фрагмента текста жирным шрифтом в меню одновременно отмечается «галочкой» строка **Bold**. Для этого необходимо создать **CheckMenuFontCharacterStyle**. Для этого перейдите в окно кода и вставьте программный код процедуры из текстового документа **CheckMenu.txt**. Метод **CheckMenuFontCharacterStyle** по

очереди проверяет стилевое оформление выделенных фрагментов, отмечая «галочками» соответствующие строки меню **CharacterStyle** или снимая со строк этого меню отметки.

37. Запустите программу и протестируйте её работу.



38. Аналогично создайте обработчики для пунктов меню **Italic**, **Underline**, **Strikeout**.

39. Подготовьте обработчики событий для строк меню **Paragraph Alignment** следующим обра-

```
private void menuFormatParagraphAlignmentLeft_Click(object sender, EventArgs e)
{
 richTextBox1.SelectionAlignment = HorizontalAlignment.Left;
}

private void menuFormatParagraphAlignmenRight_Click(object sender, EventArgs e)
{
 richTextBox1.SelectionAlignment = HorizontalAlignment.Right;
}

private void menuFormatParagraphAlignmentCenter_Click(object sender, EventArgs e)
{
 richTextBox1.SelectionAlignment = HorizontalAlignment.Center;
}
```

ЗОМ:

Все эти обработчики событий строк меню **Paragraph Alignment** изменяют значение свойства **richTextBox1.SelectionAlignment**, задающего выравнивание параграфа текста, выделенного пользователем (для выделения параграфа с целью изменения выравнивания достаточно установить в него текстовый курсор).

Создание процедур на основе событий

**Цель работы:** сформировать умения использования компонентов стандартных диалогов и системы меню, изучить основные свойства.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

Задание

**Методические указания по выполнению задания:**

1. Во многих приложениях, созданных для операционной системы **Windows**, наиболее часто используемые строки меню дублируются кнопками, расположенными на инструментальных панелях. Чтобы добавить инструментальную панель в окно приложения, перетащите мышью элемент управления **ToolStrip** вкладки **Все формы Windows Forms** панели элементов в окно проектирования формы нашего приложения. По умолчанию окно инструментальной панели появится в верхней части формы. В только что добавленной панели нет ни одной кнопки.

2. Сразу после добавления окно инструментальной панели будет расположено над окном редактора текста. Чтобы исправить это положение, щелкните правой кнопкой мыши окно редактора текста, а затем выберите из контекстного меню строку **На передний план**. В результате окна примут правильное взаимное расположение.

3. Создайте в папке проекта отдельную папку для хранения изображений инструментальной панели, назвав ее, например, **ImageList**. Далее скопируйте в созданную папку изображения, подготовленные для кнопок панели.

4. Выделите компонент **ToolStrip** и на панели свойств найдите свойство **Items**, щелкните по кнопке . Откроется окно редактора коллекций.

5. В диалоговом окне **Редактор коллекции элементов** в списке элементов выберите **Button** и нажмите кнопку **Добавить**. На панели свойств измените значение свойства **Name** данной кнопки на **создатьToolStripButton**, а размер кнопки установите равным **24**.

6. В окне свойств найдите свойство **Image**. Нажмите на кнопку , в диалоговом окне **Выбор ресурса** нажмите на кнопку **Импорт** и выберите соответствующее изображение из сохраненных ранее в папке **ImageList**.

7. Задайте комментарий-подсказку для кнопки **создатьToolStripButton**, установив свойство

**ToolStripText** равным **Создать**.

8. Аналогично создайте кнопки **открытьToolStripButton**, **сохранитьToolStripButton**, **печататьToolStripButton**.

9. Далее необходимо создать разделитель между группами кнопок, ответственных за выполнение различных групп операций. Для этого в списке элементов выберите **Separator** и нажмите кнопку **Добавить**.

10. Далее аналогично создайте кнопки **вырезатьToolStripButton**, **копироватьToolStripButton**, **вставитьToolStripButton**.

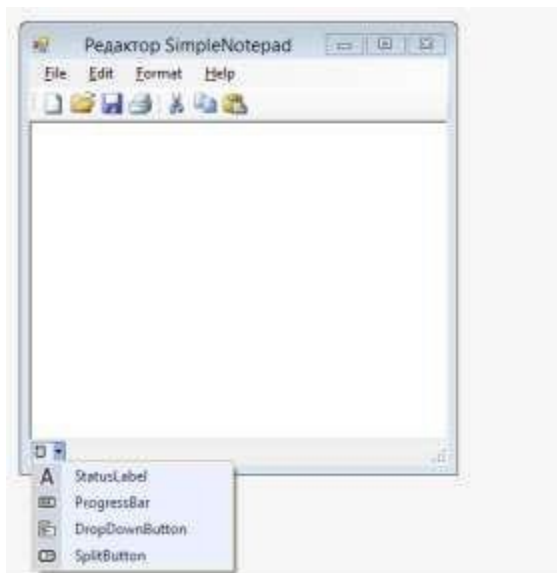
11. Теперь необходимо связать созданные ранее функции-обработчики события команд пункт меню с соответствующими кнопками панели инструментов. Выделите кнопку **создатьToolStripButton**, на панели **Свойства** перейдите на вкладку **События** , найдите событие **Click** и в списке выберите функцию **menuFileNew\_Click**. Аналогично добавьте функции для остальных кнопок панели инструментов.

12. Сохраните приложение. Запустите его и протестируйте работу кнопок панели инструментов.

Задание 2. Создание строки состояния Методические указания по выполнению задания:

1. Разместите на форме компонент **StatusStrip** со вкладки **Меню и панели инструментов** панели элементов.

2. В меню компонента **StatusStrip** выберите опцию **StatusLabel**.



3. Далее на панели **Свойства** найдите свойство **Text** и сделайте его пустым.

4. Для того чтобы отслеживать выбор строк меню необходимо предусмотреть специальный обработчик события **DropDownItemClicked**. Это событие создается строками меню, когда пользователь выбирает их при помощи мыши или клавиатуры. Создайте обработчик событий. Для этого выделите мышью меню **File** в окне дизайнера формы, а затем перейдите на вкладку **События** панели **Свойства**. В списке событий найдите событие **DropDownItemClicked** и дважды щелкните по строке рядом с ним. После этого будет создано пустое тело обработчика события. Вставьте следующий код в обработчик события:

```
private void menuFile_DropDownItemClicked(object sender, ToolStripItemClickedEventArgs e)
{
 this.UpdateStatus(e.ClickedItem);
}
```

5. Создайте пустую строку перед обработчиком данного события и опишите метод

**UpdateStatus**

следующим образом:

```
private void UpdateStatus(ToolStripItem item)
{
 if (item != null)
 {
 string msg = String.Format("{0} selected", item.Text);
 this.statusStrip1.Items[0].Text = msg;
 }
}
```

6. Выделите мышью меню **Edit** в окне дизайнера формы, а затем перейдите на вкладку **События** панели **Свойства**. В списке событий найдите событие **DropDownItemClicked** и в списке рядом с ним выберите событие **menuFile\_DropDownItemClicked**. Аналогично выполните для всех остальных пунктов меню.

7. Сохраните приложение. Запустите его и протестируйте работу строки состояния

Разработка оконного приложения с несколькими формами

**Цель работы:** сформировать практические умения по созданию простейшего приложения с несколькими экранами в среде **MIT App Inventor**

**Оборудование, технические и программные средства:** персональный компьютер, программа браузер.

Задание 1.

Используя инструменты среды **MIT App Inventor**, создать приложение, в котором при щелчке по кнопке один объект превращается в другой.

Методические указания по выполнению задания:

1. Запустите программу браузер и перейдите по ссылке **ai2.appinventor.mit.edu**.

Создайте новый проект, назовите его **Transformation**.

2. Создание приложения с несколькими экранами аналогично созданию нескольких отдельных приложений. Количество создаваемых в приложении экранов определяется разработчиком, но не может быть более 10. По умолчанию в момент создания нового проекта в нем всего один экран.

Компоненты каждого экрана создаются в режиме **Дизайнер** для этого экрана. В режиме **Блоки**

отображаются блоки, только для компонент текущего экрана.

При создании приложений с несколькими идентичными экранами, используйте функцию копирования блоков между экранами.

Навигация (переход) между экранами организуется с помощью кнопок или с помощью действий.

Каждый экран, закрывается в случае перехода на другой или возвращении на тот, с которого он был открыт.

Экраны могут обмениваться информацией путем принятия и возвращения значений, когда они открываются и закрываются. Экраны могут использовать для хранения и получения с других экранов данных только с использованием **TinyDB**, к примеру счет игры, количество набранных баллов в тесте. **TinyDB** невидимый компонент **App Inventor**, который хранит данные непосредственно в устройстве.

Компонент **TinyDB** используется внутри приложения для передачи данных между экранами, В этом их отличие от глобальных переменных, которые сохраняются в пределах одного экрана, пока приложение работает.

Приложения, созданные с помощью **App Inventor** инициализируются каждый раз заново, когда они выполняются. Это означает, что если приложение устанавливает значение переменной, а затем пользователь выходит из приложения, значение этой переменной теряется. В отличие от этого, **TinyDB** является стойким хранилище данных для приложений.

Данные, хранящиеся в **TinyDB** будут доступны каждый раз, когда приложение запускается. Каждая переменная сохраняется под собственным именем.

Каждое приложение имеет собственное хранилище данных. С помощью **TinyDB** вы не можете получить доступ к данным другого приложения. Но отсюда же следует. Что для каждого приложения имеется только одно хранилище. То есть, если сохранить переменную под именем «Х», то в данном приложении будет только одна переменная под этим именем, но она будет недоступна для других приложений.

Свойств данный компонент не имеет

Компонент **TinyDB** очень полезен, поскольку позволяет сохранить данные приложения на **Android**

устройстве. Обычно такие небольшие данные используются для сохранения настроек приложения.

После того как данные сохранены в **TinyDB** они останутся там, пока **TinyDB** не будет очищено.

3. На панели **Свойства** установите выравнивание всех объектов **Screen1** по центру, для этого выберите свойство **ВыровнятьПоГоризонтали** и задайте его значение равным **Центр**.

4. В палитре выберите компонент **Надпись** и переместите в окно **Screen1**. На панели **Свойства** поменяйте текст надписи на **Превращение в сказке**. Измените размер надписи на **28** пт.

5. На панели **Компоненты** выделите **Надпись1** и нажмите внизу панели кнопку **Переименовать**, задайте имя компонента **НадписьЗаголовок**.

6. В палитре выберите компонент **Кнопка** и переместите в окно **Screen1**. На панели **Свойства** поменяйте текст кнопки на **Превратить в комара**. Измените размер шрифта надписи на **16** пт.

7. На панели **Компоненты** выделите **Кнопка1** и нажмите внизу панели кнопку **Переименовать**, задайте имя компонента **КнопкаВКомара**.

8. Перетащите компонент **Изображение** в окно экрана. Установите в качестве источника изображения файл **gvidon.jpg**. Нажмите кнопку **Переименовать** и дайте компоненту название **ИзображениеГвидон**.

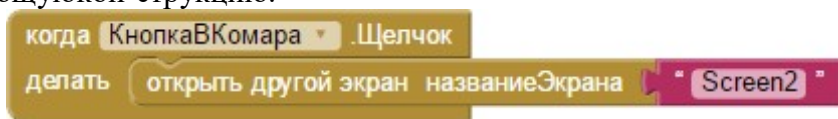


9. Нажмите кнопку **Добавить экран**. Перейдите на экран **Screen2**. Создайте по

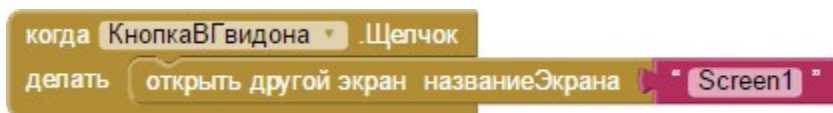


аналогии дизайн экрана.

10. Перейдите на экран **Screen1**. Перейдите на вкладку **Блоки**. Соберите из блоков следующую кон-струкцию.



11. Перейдите на экран **Screen2**. Перейдите на вкладку **Блоки**. Соберите из блоков следующую кон-струкцию.



12. Протестируйте работу.

Задание 2.

Используя инструменты среды **MIT App Inventor**, создать приложение, в котором происходит перемещение объекта с одного экрана на другой.

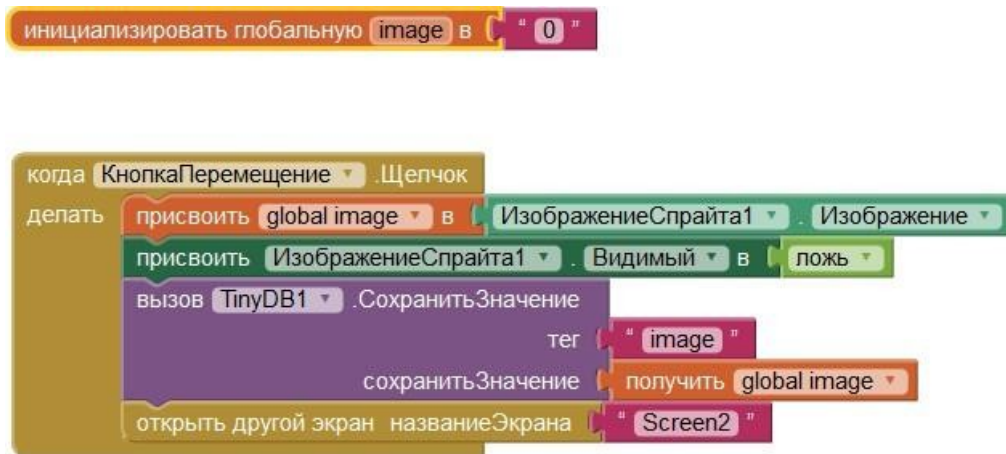
Методические указания по выполнению задания:

1. Создайте новый проект, назовите его **Moving**. Установите в параметрах **Screen1** выравнивание по центру.
2. В палитре выберите компонент **Надпись** и переместите в окно **Screen1**. На панели **Свойства** поменяйте текст надписи на **Сказочные перемещения**. Измените размер шрифта надписи на **18 пт**.
3. На панели **Компоненты** выделите **Надпись1** и нажмите внизу панели кнопку **Переименовать**, задайте имя компонента **НадписьЗаголовок**.
4. В палитре перейдите в раздел **Рисование и анимация**. Выберите компонент **Холст** и переместите его в окно **Screen1**.
5. На панели **Свойства** установите размеры холста: ширина - **300 px**, высота - **200 px**.
6. Загрузите файл изображения **ostrov.jpg**. Установите его в качестве фонового рисунка для холста.
7. В палитре в разделе **Рисование и анимация** выберите компонент **ИзображениеСпрайт** и переместите его на **Холст**.
8. Загрузите файл изображения **gvidon.png** и установите загруженное изображение как изображение спрайта.
9. В палитре выберите компонент **Кнопка** и переместите в окно **Screen1**. Выделите компонент **Кнопка** и нажмите **Переименовать**. Введите текст **КнопкаПеремещение**. Измените текст надписи на **Перенестив палаты**.



Screen1.

11. Перейдите в режим **Блоки** и соберите следующую конструкцию:



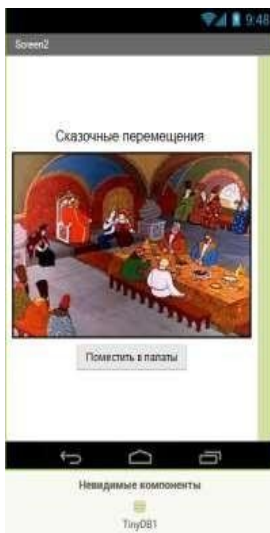
Разберем блоки конструкции более подробно. Вначале осуществляется инициализация глобальной переменной с изображением Гвидона=0, которая будет передаваться с экрана на экран, именно она и будет храниться в **TinyDB**.

Как только **Кнопка** будет нажата этой переменной необходимо присвоить **ИзображениеСпрайта** изображение Гвидона, который будет перемещаться. После перемещения изображение не должно отображаться на экране, поэтому ему присваивается параметр **Видимый – Ложь**.

Далее вызывается компонент **TinyDB**, задается изображению **ТЕГ** (имя) под которым он будет храниться на устройстве, и выбирается значение, которое нужно сохранить **GlobalImage**.

Выполняется переход на другой экран.

12. Нажмите кнопку **Добавить экран**. Перейдите на экран **Screen2**. Создайте по аналогии



дизайн экрана. Обратите внимание, что изображения Гвидона до нажатия кнопки **Поместить в палаты** нет.



Разберем блоки конструкции более подробно. Вначале также осуществляется инициализация глобальной переменной с изображением Гвидона=0, которая будет передаваться с экрана на экран, именно она и будет храниться в **TinyDB**.

Далее присваивается **ИзображениюСпрайта** изображение Гвидона, которое берётся из **TinyDB**. При получении значения - задается только **ТЕГ**.

В случае если допущена ошибку и изображение во внешнем хранилище не найдено, выдается сообщение, что оно не найдено.

14. Протестируйте работу приложения.

Задание 3.

Используя инструменты среды **MIT App Inventor**, разработайте контентное приложение с использованием меню. Для создания приложения, используйте материалы папки **SourceOcean**.

Разработка игрового приложения.

**Цель работы:** сформировать практические умения создания многооконного приложения с фреймamisредствами среды программирования **Visual Studio**.

**Оборудование, технические и программные средства:** персональный компьютер, средапрограмми-рования **Visual Studio**.

Задание 1. Создание главного окна приложенияМетодические указания по выполнению задания:

1. Запустите среду программирования **Visual Studio**. Создайте новое **Приложение Windows Forms**. Имя проекта и приложения **FramesApp**.
2. Добавьте в форму главного окна приложения меню (**MenuStrip**) и строку состояния (**StatusStrip**). В меню **File** создайте строку **Exit**, предназначенную для завершения работы приложения. Напишите об-работчик данного события.



3. Добавьте в окно нашего приложения элемент управления **TreeView**, перетащив мышью его значок из панели элементов вкладки **Все формы Windows Forms** в окно формы.

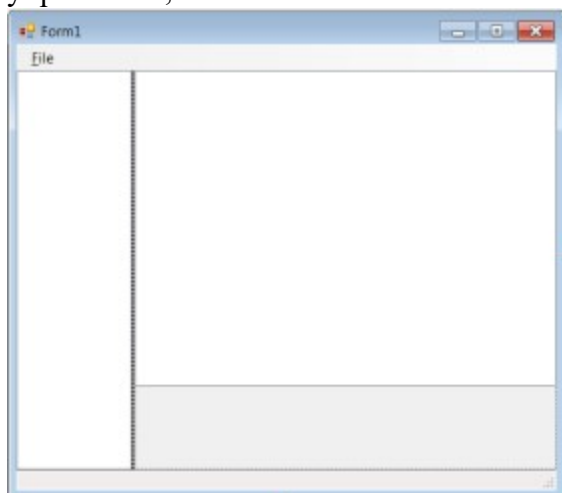
Установите значение свойства **Dock** данного элемента управления равным **Left**. В результате элемент управления **TreeView** будет выровнен по левой границе главного окна приложения. В нашем приложении окно элемента управления **TreeView** будет использовано для отображения списка дисков и каталогов.

4. Для элемента управления **TreeView** создайте разделитель. Для этого перетащите значок элемента управления **Splitter** из панели элементов вкладки **Все формы Windows Forms** в окно формы. Разделитель будет автоматически расположен справа от окна элемента управления **TreeView**.

5. Элемент управления **Splitter** имеет свойства, которые можно менять в процессе разработки приложения, а также динамически во время его работы. Самое важное свойство разделителя **Splitter** — это свойство **Dock**, задающее его расположение. По умолчанию значение этого свойства равно **Left**, благодаря чему разделитель разместился слева от окна дерева **TreeView**. При необходимости можно изменять расположение разделителя с помощью окна редактирования, аналогичного окну.

6. Перетащите значок элемента управления **Panel** из панели элементов вкладки **Все формы Windows Forms** в окно формы, а затем установите значение свойства **Dock** этой панели равным **Fill**. В результате наша панель будет расположена справа от разделителя и займет всю правую часть окна приложения.

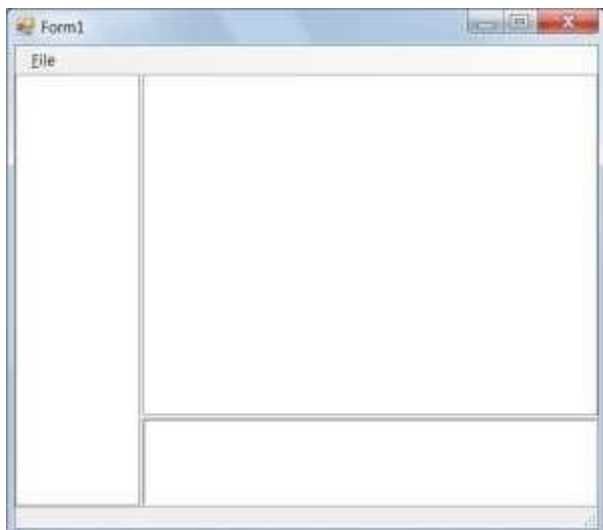
7. В верхнем окне будет находиться элемент управления **ListView**, показывающий список файлов и каталогов. Добавьте в окно приложения элемент управления **ListView**, перетащив его значок из панели элементов вкладки **Все формы Windows Forms** в окно формы. После установки значения свойства **Dock**, равным **Top**, и увеличения высоты окна этого элемента управления, главное окно нашего приложения примет следующий вид.



8. Добавьте в правую часть главного окна горизонтальный разделитель **Splitter**. Перетащите его пиктограмму из панели элементов вкладки **Все формы Windows Forms** в окно формы, а затем установите значение свойства **Dock** добавленного разделителя, равным **Top**. В результате разделитель будет расположен непосредственно под окном элемента управления **ListView**.

9. В нижней части окна приложения будет находиться элемент управления **RichTextBox**, предназначенный для отображения дополнительной информации. Перетащите значок элемента управления **RichTextBox** из панели элементов вкладки **Все формы Windows Forms** в окно формы и установите значение свойства **Dock** равным **Fill**.

10. Протестируйте работу приложения. Сохраните полученный проект. Не написав ни единой строчки кода, вы создали простейшее приложение с действующим многооконным



интерфейсом поль-зователя.

файлами, подключите пространство имен **System.IO**.

```
using System.IO;
```

1. Перейдите в окно редактирования кода и создайте метод **DriveTreeInit**:

```
public void DriveTreeInit()
{
 string[] drivesArray = Directory.GetLogicalDrives();

 treeView1.BeginUpdate();
 treeView1.Nodes.Clear();

 foreach(string s in drivesArray)
 {
 TreeNode drive = new TreeNode(s, 0, 0);
 treeView1.Nodes.Add(drive);

 GetDirs(drive);
 }
 treeView1.EndUpdate();
}
```

В самом начале своей работы этот метод получает список логических дисковых устройств, установленных в системе, и сохраняет его в массиве **drivesArray**. Далее метод **DriveTreeInit** вызывает метод **treeView1.BeginUpdate**. Этот метод временно блокирует перерисовку окна дерева до тех пор, пока не будет вызван метод **treeView1.EndUpdate**. Пара этих методов используется в том случае, когда нужно добавить, удалить или изменить большое количество элементов дерева. Если не заблокировать перерисовку окна, на обновление дерева уйдет слишком много времени.

В классе **TreeView** определено свойство **Nodes**, хранящее все узлы дерева. Перед тем как приступить к заполнению дерева, метод **DriveTreeInit** удаляет все узлы, вызывая для этого метод **treeView1.Nodes.Clear**.

Заполнение дерева происходит в цикле. Массив **drivesArray** содержит массив текстовых строк с обозначениями логических устройств, установленных в системе. Для каждого такого устройства в цикле создается объект класса **TreeNode** — узел дерева. В качестве первого параметра конструктору передается текстовая строка названия устройства. Созданный узел добавляется в набор узлов дерева с помощью метода **treeView1.Nodes.Add**. В качестве параметра этому методу передается ссылка на объект **TreeNode**, т.е. на добавляемый узел. Далее вызывается метод **GetDirs**, добавляющий в дерево список содержимого корневого каталога текущего дискового устройства.

```
public void GetDirs(TreeNode node)
{
 DirectoryInfo[] diArray;
 node.Nodes.Clear();

 string fullPath = node.FullPath;
 DirectoryInfo di = new DirectoryInfo(fullPath);

 try
 {
 diArray = di.GetDirectories();
 }
 catch
 {
 return;
 }

 foreach(DirectoryInfo dirinfo in diArray)
 {
 TreeNode dir = new TreeNode(dirinfo.Name, 0, 0);
 node.Nodes.Add(dir);
 }
}
```

Первым делом метод **GetDirs** удаляет все элементы из текущего узла, вызывая для этого метод

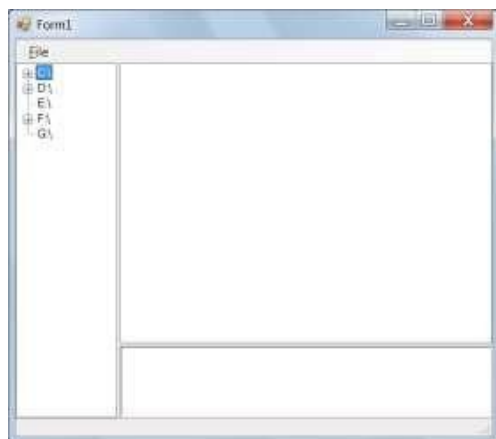
`node.Nodes.Clear`.

Далее метод переписывает в переменную **fullPath** типа **string** полный путь **node.FullPath** к узлу дерева. Эта строка получается объединением текстовых строк всех родительских узлов. Если корневой узел хранит текстовую строку обозначения логического диска, то после выполнения этой процедуры в переменной **fullPath** будет храниться полный путь к файлу или каталогу.

Далее для получения содержимого каталога, на который ссылается переменная **fullPath**, используется метод **GetDirectories**. При возникновении исключения программа просто возвращает управление, не выполняя над узлом дерева никаких функций.

В том случае если содержимое каталога было успешно получено, оно сохраняется в массиве **diArray**. Далее содержимое массива **diArray** используется для заполнения узла дерева содержимым каталога.

2. Запустите программу на исполнение. В окне дерева появится список дисковых



устройств. Рас-крываются ли узлы дерева при щелчке мышью?

при помощи клавиатуры, нужно обрабатывать событие **BeforeExpand**. Для создания обработчика этого события выделите дерево в окне дизайнера формы, а затем откройте вкладку событий. Далее в поле **BeforeExpand** ведите имя обработчика событий **treeView1\_OnBeforeExpand**. После этого добавьте в этот обработчик событий следующий код:

```
private void treeView1_OnBeforeExpand(object sender, TreeViewCancelEventArgs e)
{
 treeView1.BeginUpdate();

 foreach (TreeNode node in e.Node.Nodes)
 {
 GetDirs(node);
 }

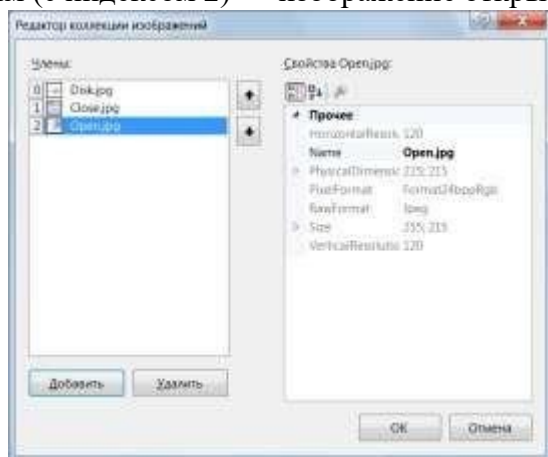
 treeView1.EndUpdate();
}
```

Событие **BeforeExpand** возникает при попытке пользователя раскрыть узел дерева. В этом случае наш обработчик заполняет открываемый узел при помощи рассмотренного ранее ме-тода **GetDirs**. Ссылка на узел извлекается из поля **e.Node.Nodes**, передаваемого обработчику событий в качестве параметра.

3. Запустите приложение. Раскрываются ли узлы дерева при щелчке мышью теперь?

4. Вы можете улучшить внешний вид дерева, если добавите к его узлам значки дисковых устройств и каталогов (папок). В папке проекта создайте папку **ImageList**. Подготовьте эти значки, используя любой графический редактор. Скопируйте эти значки в папку **ImageList**. Чтобы подключить флажки к проекту, выберите из меню проект системы **Microsoft Visual Studio** строку **Добавить существующий элемент**, а затем добавьте файлы значков при помощи появившегося на экране диалогового окна.

5. Чтобы использовать изображения в дереве, их необходимо объединить в список класса **ImageList**. Добавьте этот список в приложение, перетащив значок компонента **ImageList** в окно проектирования формы из панели элементов вкладки **Все формы Windows Forms**. Выделив элемент **imageList1** левой клавишей мыши, отредактируйте его свойство **Images**. Для редактирования будет открыто окно **Редактор коллекций изображений**. Воспользуйтесь кнопкой **Добавить** для добавления в список файлов изображений, скопированных ранее в каталог нашего приложения. Изображения следует разместить в следующем порядке: первым (с индексом 0) должно идти изображение для диска, вторым (с индексом 1) — изображение закрытой папки, и третьим (с индексом 2) — изображение открытой папки.



6. Создав и заполнив список изображений, подключите его к дереву просмотра дисков и каталогов. Для этого отредактируйте свойство **ImageList** элемента управления **treeView1**,

присвоив ему ссылку на список изображений **imageList1**.

7. Для отображения значков в узлах дерева необходимо изменить исходный текст методов **DriveTreeInit** и **GetDirs**. Первый из этих методов инициализирует дерево, а второй — добавляет к узлу дерева списоккаталогов. Обратите внимание на конструктор класса **TreeNode**, создающий узлы дерева внутри тела цикла **foreach**. Первый параметр задает текст надписи для узла дерева. Если к элементу управления **TreeView** подключен список изображений, то второй и третий параметры конструктора класса **TreeNode** задают индексы изображений для узла дерева. При этом второй параметр определяетизображение невыделенного узла дерева, а третий — выделенного. Что касается метода **DriveTreeInit**, то расположенный в нем конструктор создает узлы, отображающий только дисковые устройства. В любом состоянии (как выделенном, так и невыделенном) нам необходимо отображать один и тот же

значок дискового устройства, имеющий в нашем случае индекс 0. Поэтому второй и третий

```
public void DriveTreeInit()
{
 string[] drivesArray = Directory.GetLogicalDrives();

 treeView1.BeginUpdate();
 treeView1.Nodes.Clear();

 foreach(string s in drivesArray)
 {
 TreeNode drive = new TreeNode(s, 0, 0);
 treeView1.Nodes.Add(drive);

 GetDirs(drive);
 }
 treeView1.EndUpdate();
}
```

параметрыконструктора передают нулевые значения.

8. В методе **GetDirs** конструктору класса **TreeNode**, размещенному внутри оператора цикла **foreach**, черезвторой и третий параметры передаются индексы значков закрытой и открытой

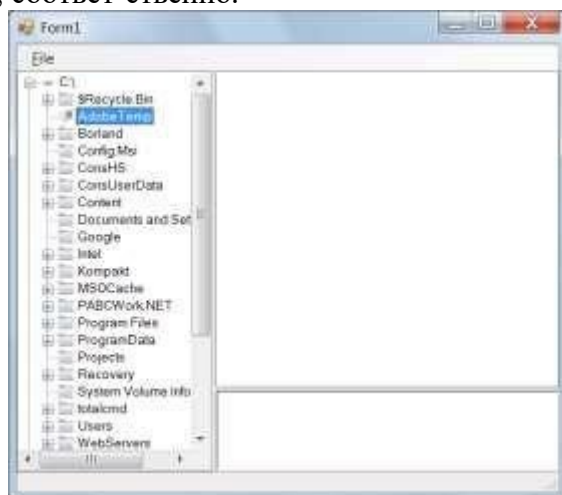
```
public void GetDirs(TreeNode node)
{
 DirectoryInfo[] diArray;
 node.Nodes.Clear();

 string fullPath = node.FullPath;
 DirectoryInfo di = new DirectoryInfo(fullPath);

 try
 {
 diArray = di.GetDirectories();
 }
 catch
 {
 return;
 }

 foreach(DirectoryInfo dirinfo in diArray)
 {
 TreeNode dir = new TreeNode(dirinfo.Name, 1, 2);
 node.Nodes.Add(dir);
 }
}
```

папки, соответственно.



9. Выделите элемент управления **TreeView** на панели Свойства найдите свойство **CheckBoxes** и установите его значение **True**. Теперь узлы дерева будут иметь индивидуальные флажки. Отметив все или некоторые узлы дерева флажками, можно выполнять над соответствующими объектами групповые операции.

10. Выделите элемент управления **TreeView** на панели Свойства найдите свойство **LabelEdit** и установите его значение **True**. Теперь пользователь может редактировать текстовые надписи, расположенные около узлов дерева.

Тестирование, отладка приложения

**Цель работы:** сформировать практические умения создания многооконного приложения с фреймами средствами среды программирования **Visual Studio**.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

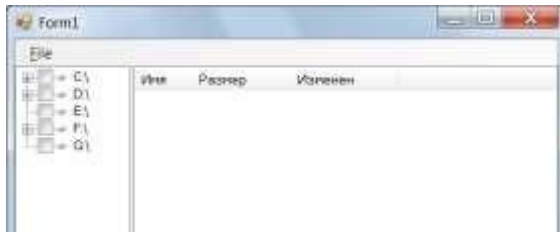
Задание 3. Создание методов для элемента управления **ListView** Методические указания по выполнению задания:

1. Выделите список **ListView** в окне дизайнера форм и установите для свойства **View** значение **Details**. При необходимости программа сможет динамически переключать режимы отображения, изменяя значение свойства **View** во время своей работы.

2. Поскольку в приложении будет отображаться список в виде таблицы, то необходимо создать столбцы таблицы и определить их атрибуты. Для этого необходимо отредактировать свойство **Columns**. Это делается при помощи **Редактор коллекции ColumnHeader**.

3. С помощью кнопки **Добавить** добавьте в таблицу три столбца, а затем настройте свойства **Text**, **TextAlign** и **Width** этих столбцов. Свойство **Text** задает название столбца, отображаемого в верхней части списка **ListView**. Первый столбец (с индексом 0) должен называться **Имя**, второй — **Размер**, а третий — **Изменен**. Свойство **TextAlign** задает выравнивание заголовка столбца. По умолчанию это свойство имеет значение **Left**, задающее выравнивание по левой границе. С помощью свойства **Width** можно указать начальную ширину столбца в пикселях. После отображения списка пользователь сможет изменять ширину столбцов при помощи мыши.

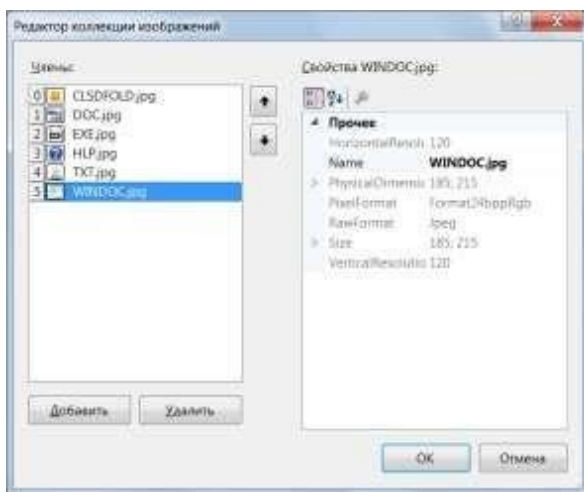
4. Запустите приложение на исполнение. Теперь в правом верхнем фрейме приложения появится пустой список с заголовками.



5. Элемент управления **ListView** может отображать содержимое своего окна в четырех различных режимах, задаваемых при помощи свойства **View**. В зависимости от выбранного режима, элементы

списка могут снабжаться значками маленького или большого размера. В приложении **FramesApp** будут использованы шесть значков для отображения папок и файлов различных типов (файлы исполнимых программ, файлов справочной системы, текстовых файлов, файлов документов и пр.). Скопируйте файлы значков в папку **ImageList** созданную ранее.

6. Для работы с этими значками потребуется еще один список изображений **ImageList**. Перетащите его значок с панели элементов в окно дизайнера форм. Новый список будет



иметь идентификатор **imageList2**. Отредактируйте данный список.

7. В нашем приложении список **ListView** отображается только в одном режиме, а именно, как детализированная таблица. Такая таблица снабжается значками только маленького размера. Если же Ваше приложение будет использовать режим **LargeIcon**, необходимо подготовить дополнительный список **ImageList**, добавив в него файлы значков большого размера. Подключите подготовленный список к элементу управления **ListView**. Для этого присвойте свойствам **SmallImageList**, **LargeImageList** значение **imageList2**.

8. В общем случае список **ListView** содержит элементы, каждый из которых имеет дополнительные элементы более низкого уровня, которые называются атрибутами. Сам элемент представляется текстовой строкой в первом столбце списка, отображаемого в виде детализированной таблицы. Значения остальных атрибутов отображаются в остальных столбцах таблицы. Теперь необходимо наполнить список **ListView** функциональностью. Когда пользователь выделяет диск или каталог в дереве просмотра **TreeView**, создается событие **AfterSelect**. Обработчик этого события должен определить, какой диск или какой каталог был выделен, а затем наполнить окно списка **ListView** именами каталогов и файлов, расположенных на этом диске или в этом каталоге. Создайте обработчик события **AfterSelect**. Предварительно создайте поле **fullPath** в классе **Form1**, вставив следующую строку кода: `public string fullPath;`

Код обработчика события скопируйте из файла **AfterSelect.txt**.

Прежде всего, данный метод извлекает ссылку на узел дерева, выделенный пользователем, из свойства **Node** параметра обработчика событий **treeView1\_OnAfterSelect**. Далее, полный путь к выделенному узлу записывается в поле **fullPath** класса **string**.

Так как наше дерево содержит только обозначения дисков и каталогов, то это будет путь либо к корневоуматалогу диска, либо к одному из подкаталогов. Далее мы создаем объект класса **DirectoryInfo** и получаем списки всех файлов и каталогов, располагающихся в каталоге, выделенном пользователем в дереве. Для выполнения этих операций применяются методы **GetFiles** и **GetDirectories**. Перечень файлов обработчик события сохраняет в массиве **fiArray**, а перечень каталогов — в массиве **diArray**.

Наполнение списка именами каталогов выполняется в цикле **foreach**. С помощью конструктора класса **ListViewItem** мы создаем элемент списка, а затем задаем значения атрибутов этого элемента. Длина каталогов считается равной нулю, а время последнего изменения каталога извлекается при помощи метода **LastWriteTime** и преобразуется в текстовую строку методом **ToString**.

В свойство **lvi.ImageIndex** записывается нулевое значение — индекс значка в списке изображений

**imageList2** с изображением закрытой папки.

После заполнения всех атрибутов элемента списка этот элемент добавляется в список методом

**listView1.Items.Add**.

На следующем этапе в список добавляются имена файлов, расположенных в выделенном каталоге. Эти имена тоже добавляются в цикле

Размер очередного файла мы получаем с помощью свойства **Length**, а дату последнего изменения — с помощью свойства **LastWriteTime** (как и для каталогов).

Расширение имени файла извлекается из полного имени методом **Path.GetExtension**, а затем все его буквы преобразуется в прописные методом **ToLower**. Непосредственный выбор значка выполняется при помощи оператора **switch**. Подготовленный элемент списка, описывающий текущий файл, добавляется в список методом **listView1.Items.Add**.

9. Разработайте обработчик события **ItemActivate** к элементу управления **ListView**, для того чтобы было можно отображать в окне текстового редактора содержимое текстовых файлов, отображаемых в правом верхнем фрейме. Это событие возникает, когда пользователь дважды щелкает строку в окне списка **ListView**. Для того, чтобы в строке состояния отображалась информация об имени файла добавьте на строку состояния **StatusLabel**. Исходный текст обработчика события **ItemActivate** имеет следующий вид:

```

private void OnItemActivate(object sender, EventArgs e)
{
 foreach(ListViewItem lvi in listView1.SelectedItems)
 {
 string ext = Path.GetExtension(lvi.Text).ToLower();
 if (ext==".txt"||ext==".htm"||ext==".html")
 {
 try
 {
 richTextBox1.LoadFile(Path.Combine(fullPath, lvi.Text),
 RichTextBoxStreamType.PlainText);

 StatusLabel1.Text = lvi.Text;
 }
 catch
 {
 return;
 }
 }
 else if (ext==".rtf")
 {
 try
 {
 richTextBox1.LoadFile(Path.Combine(fullPath, lvi.Text),
 RichTextBoxStreamType.RichText);

 StatusLabel1.Text = lvi.Text;
 }
 catch
 {
 return;
 }
 }
 }
}

```

В качестве первого параметра методу передается полный путь к файлу, полученный комбинированием полного пути каталога **fullPath**, выделенного в дереве, и имени файла **lvi.Text**, выделенного в списке **ListView**.

Через второй параметр методу **LoadFile** передается тип загружаемого документа, заданный как **RichTextBoxStreamType.PlainText** (т.е. текстовый документ). Дополнительно имя документа отображается в строке состояния главного окна приложения, для чего это имя переписывается из свойства **lvi.Text** в свойство **StatusLabel1.Text**.

## Программирование приложений

**Цель работы:** научиться создавать проекты с использованием класса.

**Оборудование, технические и программные средства:** персональный компьютер, программа браузер.

```

void DayStud::CalcStip()
{if (rait>=90) stip=300;
 else if (rait>=76)
 stip=200;
 else stip=0;
}
void DayStud::WriteSt()
{ Student::WriteSt();
 cout<<"Stipendiya"<<stip;
}
//пример использования определенных выше классов
int main()
{ const int m=5; //будем работать с 5-ю студентами
 int i;
 DayStud gr[m];
 for(i=0;i<m;i++)
 gr[i].ReadSt(); //вводим информацию о каждом студенте
 for(i=0;i<m;i++)
 { cout<<"Examen"<<i+1<<" student";
 gr[i].Exam(); //проводим экзамены (вводим информацию о баллах, полученных
 //каждым студентом на экзаменах
 }
 for(i=0;i<m;i++)
 gr[i].CalcRait(); //вычисляем рейтинг каждого студента
 for(i=0;i<m;i++)
 gr[i].CalcStip(); //вычисляем стипендию каждого из студентов
 for(i=0;i<m;i++)
 gr[i].WriteSt(); //выводим информацию о каждом студенте на экран
 return 0;
}

```

#### Контрольные вопросы:

1. Укажите имя базового класса в проекте C34PR1.
2. Перечислите свойства базового класса проекта C34PR1.
3. Перечислите методы базового класса проекта C34PR1.
4. Перечислите классы-потомки и укажите для каждого из них ключ доступа, используемый при наследовании для проекта C34PR1.
5. Перечислите свойства классов-потомков проекта C34PR1.
6. Перечислите методы, вызываемые от унаследованного класса проекта C34PR1.

**ФОС для текущего контроля****знать:**

- Что такое алгоритм и какие свойства он должен иметь?
- Какие существуют способы описания алгоритмов?
- В чём заключается принцип дискретности алгоритма?
- Как оценивается сложность алгоритма?
- Какие существуют базовые алгоритмические конструкции?
- Как реализуется последовательность действий в алгоритме?
- В чём особенность ветвления в алгоритмах?
- Как работает циклическая конструкция?
- Что такое структура данных и зачем она нужна?
- Какие существуют основные типы данных?
- Как устроены массивы и как с ними работать?
- В чём особенность работы со строками?
- Что такое язык программирования?
- Какие существуют типы языков программирования?
- Как происходит трансляция программы?
- В чём разница между компиляцией и интерпретацией?
- Что такое подпрограмма и зачем она нужна?
- Как передаются параметры в функции?
- В чём особенность рекурсивных функций?
- Как работает механизм возврата значений?
- Что такое объект и класс?
- Как реализуется инкапсуляция?
- В чём особенность наследования?
- Как работает полиморфизм?
- Какие существуют правила записи программы?
- Как работают операторы присваивания?
- В чём особенность арифметических операций?
- Как работают логические операторы?
- Как осуществляется файловый ввод-вывод?
- Какие существуют режимы работы с файлами?
- Как организуется последовательный доступ?
- В чём особенность прямого доступа к файлам?
- Что такое ошибка в программе?
- Как происходит отладка программы?
- Какие существуют методы тестирования?

- Как составляется тест-кейс?
- Какие существуют алгоритмы сортировки?
- В чём особенность быстрой сортировки?
- Как работает сортировка слиянием?
- Как оценивается эффективность алгоритмов сортировки?
- Какие существуют методы поиска?
- Как работает бинарный поиск?
- В чём особенность хеш-поиска?
- Как оценивается сложность алгоритмов поиска?
- Что такое связный список?
- Как устроены стеки и очереди?
- В чём особенность деревьев?
- Как работают графы?
- Что такое оптимизация программы?
- Как оптимизировать использование памяти?
- В чём особенность оптимизации циклов?
- Как ускорить работу программы?
- Какие существуют стандарты кодирования?
- Как оформляется программный код?
- В чём особенность документирования?
- Как составляются комментарии?
- Какие существуют современные подходы к программированию?
- В чём особенность функционального программирования?
- Как работает параллельное программирование?
- Какие существуют современные инструменты разработки?

#### **уметь:**

- Разработать алгоритм вычисления факториала числа
- Реализовать алгоритм поиска максимального элемента в массиве
- Написать программу для вычисления суммы элементов массива
- Создать алгоритм проверки числа на простоту
- Составить блок-схему решения задачи
- Написать программу определения високосного года
- Реализовать проверку принадлежности точки к области на плоскости
- Создать калькулятор с выбором операций
- Разработать программу классификации треугольников
- Написать алгоритм определения четности числа
- Реализовать вывод таблицы умножения
- Написать программу вычисления суммы числового ряда

- Создать алгоритм подсчета количества цифр в числе
- Разработать программу генерации последовательности Фибоначчи
- Написать алгоритм проверки палиндрома
- Реализовать сортировку массива методом пузырька
- Написать программу поиска элемента в массиве
- Создать алгоритм удаления дубликатов из массива
- Разработать программу транспонирования матрицы
- Написать алгоритм умножения матриц
- Реализовать подсчет частоты букв в тексте
- Написать программу замены подстроки
- Создать алгоритм проверки анаграмм
- Разработать программу подсчета слов в тексте
- Написать алгоритм шифрования текста
- Создать библиотеку математических функций
- Написать рекурсивную функцию вычисления чисел Фибоначчи
- Разработать процедуру сортировки массива
- Создать функцию проверки корректности email
- Написать процедуру обработки ошибок
- Разработать класс для работы с геометрическими фигурами
- Создать иерархию классов для животных
- Написать программу с использованием наследования
- Разработать систему классов для библиотеки
- Создать приложение с применением полиморфизма
- Реализовать чтение и запись текстовых файлов
- Написать программу анализа данных из файла
- Создать приложение для работы с бинарными файлами
- Разработать программу архивации данных
- Написать алгоритм поиска в файлах
- Реализовать стек на основе массива
- Создать очередь с приоритетами
- Написать программу работы со связным списком
- Разработать бинарное дерево поиска
- Создать хеш-таблицу
- Реализовать быструю сортировку
- Написать программу сортировки слиянием
- Создать алгоритм пирамидальной сортировки
- Разработать программу подсчета инверсий
- Написать оптимизированный алгоритм сортировки
- Реализовать бинарный поиск

- Написать программу поиска в глубину
- Создать алгоритм поиска в ширину
- Разработать программу A\* поиска
- Написать алгоритм поиска подстроки
- Реализовать обход графа в глубину
- Написать программу поиска кратчайшего пути
- Создать алгоритм построения минимального остовного дерева
- Разработать программу проверки связности графа
- Написать алгоритм топологической сортировки
- Реализовать мемоизацию для рекурсивных функций
- Написать программу с использованием динамического

программирования

- Создать оптимизированный алгоритм поиска подстроки
- Разработать эффективный алгоритм обработки больших данных
- Написать программу с минимизацией использования памяти
- Реализовать обработку исключений
- Написать программу с валидацией входных данных
- Создать систему журналирования ошибок
- Разработать механизм восстановления после сбоев
- Написать программу с проверкой граничных условий
- Создать калькулятор с графическим интерфейсом
- Разработать систему учета товаров
- Написать программу для анализа данных
- Создать игру с использованием алгоритмов ИИ
- Разработать приложение для обработки изображений.

## Задания для самостоятельной работы

## Теоретические задания

1. Что такое алгоритм? Назовите его основные свойства.
2. В чём разница между алгоритмом и программой?
3. Что такое переменная и константа в программировании?
4. Какие типы данных вы знаете? Приведите примеры.
5. Что такое выражение и оператор в языке программирования?
6. Для чего используются комментарии в программе?
7. Что такое блок-схема? Какие основные элементы блок-схем вы знаете?
8. В чём разница между синтаксисом и семантикой языка программирования?
9. Что такое отладка и тестирование программы?
10. Какие этапы включает процесс разработки программы?
11. Что такое линейный алгоритм? Приведите пример.
12. Что такое ветвящийся алгоритм? Какие операторы используются для его реализации?
13. В чём разница между операторами IF и CASE?
14. Что такое цикл? Какие виды циклов вы знаете?
15. В чём отличие цикла с предусловием от цикла с постусловием?
16. Как работает цикл FOR? Для чего он применяется?
17. Что такое вложенные циклы? Приведите пример.
18. Как можно досрочно выйти из цикла?
19. Что такое бесконечный цикл и как его избежать?
20. Как организовать многократное выполнение определённого участка программы?
21. Что такое массив? Как объявить одномерный массив?
22. Как обратиться к элементу массива?
23. Что такое многомерный массив? Приведите пример.
24. Как заполнить массив данными с клавиатуры?
25. Как вывести элементы массива на экран?
26. Как найти сумму элементов массива?
27. Как найти максимальный (минимальный) элемент массива?
28. Что такое сортировка массива? Какие методы сортировки вы знаете?
29. В чём суть сортировки пузырьком?
30. Как реализовать поиск элемента в массиве?
31. Что такое процедура и функция? В чём их отличие?
32. Как передать параметры в процедуру или функцию?
33. Что такое локальные и глобальные переменные?
34. Для чего используются подпрограммы в программах?
35. Как описать функцию в Pascal?
36. Что такое рекурсия? Приведите пример.
37. Какие типы файлов вы знаете?

38. Как описать файловую переменную?
39. Как открыть файл для чтения, записи, добавления данных?
40. Как закрыть файл после работы с ним?
41. Как считать данные из файла в массив?
42. Как записать массив в файл?
43. Как проверить, существует ли файл на диске?
44. Для чего служит раздел VAR в программе на Pascal?
45. Что такое оператор присваивания? Как он записывается в Pascal?
46. Какие операторы ввода/вывода используются в Pascal?
47. Что такое строковый тип данных? Как с ним работать?
48. Какие стандартные математические функции есть в Pascal?
49. Что такое логические операции AND, OR, NOT, XOR? Приведите примеры.
50. Какие зарезервированные слова нельзя использовать в качестве имён переменных?

### Практические задания

- 1 Даны три целых числа:  $A$ ,  $B$ ,  $C$ . Проверить истинность высказывания: «Ровно одно из чисел  $A$ ,  $B$ ,  $C$  положительное».
- 2 Даны три целых числа:  $A$ ,  $B$ ,  $C$ . Проверить истинность высказывания: «Ровно два из чисел  $A$ ,  $B$ ,  $C$  являются положительными».
- 3 Проверить истинность высказывания: «Среди трех данных целых чисел есть хотя бы одна пара совпадающих».
- 4 Проверить истинность высказывания: «Среди трех данных целых чисел есть хотя бы одна пара взаимно противоположных».
- 5 Дано трехзначное число. Проверить истинность высказывания: «Все цифры данного числа различны».
- 6 Дано трехзначное число. Проверить истинность высказывания: «Цифры данного числа образуют возрастающую последовательность».
- 7 Дано трехзначное число. Проверить истинность высказывания: «Цифры данного числа образуют убывающую последовательность».
- 8 Дано целое число. Если оно является положительным, то прибавить к нему 1; если отрицательным, то вычесть из него 2; если нулевым, то заменить его на 10. Вывести полученное число.
- 9 Даны три целых числа. Найти количество положительных чисел в исходном наборе.
- 10 Даны три целых числа. Найти количество отрицательных чисел в исходном наборе.
- 11 Дано действительное число  $X$  и целое число  $N$ . Вывести  $N$  первых членов геометрической прогрессии и найти сумму ее первых членов, если первый член  $X$ , а знаменатель  $(-0,38)$ .
- 12 Дано действительное число  $X$  и целое число  $N$ . Вывести  $N$  первых членов арифметической прогрессии и найти сумму ее  $N$  первых членов, если первый член  $X$ , а разность 4,98.

13 Дано действительное число X и целое число N. Вывести N первых членов геометрической прогрессии и найти сумму ее первых членов, если первый член X, а знаменатель  $(-8,38)$ .

### Задачи

1. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.
2. Определить силу притяжения F между телами массы m1 и m2, находящимися на расстоянии r друг от друга. ( $F = -G \cdot m1 \cdot m2 / r^2$ , где  $G = 6,67 \cdot 10^{-8} \text{ см}^3 / (\text{г} \cdot \text{сек}^2)$  – гравитационная постоянная)
3. Даны действительные числа x, y, z. Вычислить:

$$\text{а) } a = \ln \left( y - \sqrt{|x|} \left( x - \frac{y}{z + \frac{x^2}{4}} \right) \right) \quad \text{б) } b = y + \frac{x}{y^2 + \frac{x^2}{y + \frac{x^3}{3}}}$$

4. Даны 2 действительных числа. Заменить первое число нулем, если оно больше или равно второму. И оставить числа без изменения в противном случае.
5. Даны 3 действительных числа. Возвести в квадрат те из них, значения которых неотрицательны.
6. Даны действительные числа x, y. Если x и y отрицательны, то каждое из них заменить его модулем; если отрицательно только одно из них, то оба значения увеличить на 0.5. В остальных случаях значения оставить без изменения.
7. Дан ряд  $a_k = \frac{k}{k^2 + 1}$ . Найти сумму всех членов ряда не меньших числа  $\varepsilon$ .
8. Вычислить  $(1 + \sin 0.1)(1 + \sin 0.2) \dots (1 + \sin 10)$ .
9. Вычислите сумму квадратов нечетных чисел последовательности от 11 до 99.
10. Вывести на печать числа, обратные квадратам первых ста натуральных чисел, и посчитать их произведение.
11. Составить блок-схему и программу вычисления суммы  $\sum_{n=1}^{50} \frac{n+1}{n}$ .

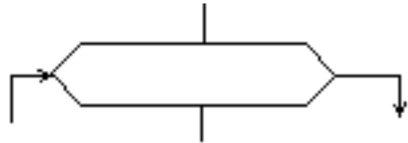
### Перечень заданий закрытого типа

Задание № 1. Как называется принцип моделирования требуемых атрибутов и взаимодействий сущностей в виде классов для определения основного представления системы?

- а) инкапсуляция.
- б) наследование.
- в) абстракция.

г) полиморфизм.

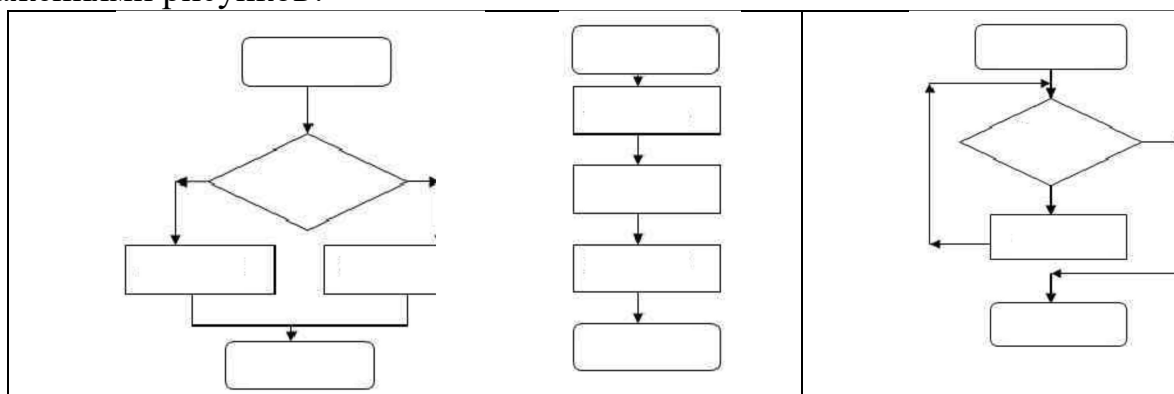
Задание № 2. Что из нижеперечисленного записывается внутри данного блочного символа



при составлении алгоритма решения задачи?

- а) ввод входных данных;
- б) комментарий;
- в) условие;
- г) вычислительное действие;
- д) начальное значение, граничное условие и шаг изменения значения параметра для повторения.

Задание № 3. Сопоставьте названия алгоритмических конструкций с изображениями рисунков?



- а) линейная;
- б) циклическая;
- в) разветвляющаяся.

Задание № 4. Сопоставьте какая алгоритмическая конструкция подходит для каждой из приведенных постановок задач?

| 1                                                 | 2                                                                                             | 3                                                                                                                                                     |
|---------------------------------------------------|-----------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Вычислить произведение и сумму чисел от 10 до 50. | Вычислить значение выражения для $x=8$ и $y=16$ :<br>$z = 1 + \frac{x^2}{2} + \frac{y^3}{4}.$ | Даны два действительных числа. Заменить первое число нулем, если оно меньше второго или равно ему, и оставить числа без изменения в противном случае. |

- а) линейная;
- б) циклическая;
- в) разветвляющаяся.

Задание № 5. Установите правильную последовательность шагов алгоритма решения квадратного уравнения.

а) Если ( $D > 0$ ), вычислить  $X1 = (-B + \sqrt{D})/(2*A)$ ,  $X2 = (-B - \sqrt{D})/(2*A)$ , вывести сообщение "Два корня,  $X1 =$ ",  $X1$ , "  $X2 =$ ",  $X2$ .

б) Вычислить дискриминант  $D = B^2 - 4*A*C$ .

в) Если ( $D < 0$ ), вывести сообщение "Уравнение действительных решений не имеет".

г) Если ( $D = 0$ ), вычислить  $X = -B/(2*A)$ , вывести сообщение "Один корень,  $X =$ ",  $X$ .

д) Получить коэффициенты уравнения -  $A$ ,  $B$ ,  $C$ .

### **Перечень заданий открытого типа**

Задание №1. На каком этапе разработки программного обеспечения выбирается форма хранения данных и составляются все необходимые формулы?

Задание №2. Какой принцип объектно-ориентированного программирования формулируется как скрытие внутреннего состояния и функций объекта и предоставление доступа только через открытый набор функций?

Задание №3. Как называется набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата?

Задание №4. Дополните определение: «Описание структуры алгоритма на естественном, частично формализованном языке, позволяющее выявить основные этапы решения задачи, перед точной его записью на языке программирования называется ...».

Задание №5. Дополните определение: «Совокупность действий, направленных на устранение ошибок в программах, начиная с момента обнаружения фактов ошибочной работы программы и завершая устранением причин их возникновения называется ...?»

### ***Формируемая компетенция: ПК 1.2***

### **Перечень заданий закрытого типа**

Задание № 1. Какая результатная информация отобразится на экране после выполнения программы, если на вход программы подать  $a=10$ ,  $b=20$ ?

```

a = int(input())
b = int(input())
if a < b:
 print(a)
else:
 print(b)

```

- а) 10;
- б) 20;
- в) 30;
- г) -10.

Задание № 2. Укажите, значение переменной *i* после выполнения данного фрагмента программы:

```

i=1
while i<=10:
 i=i+1

```

- а) 9;
- б) 10;
- в) 11;
- г) 2.

Задание № 3. Сопоставьте постановки задач и предложенные программные решения.

| 1)                                                                                  | 2)                                                                                                                  | 3)                                                                                                                                                                                                              |
|-------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> n = int(input()) s = 0 for i in range(1,n + 1, 1):     s += i print(s) </pre> | <pre> num = 75869 count = 0 while num != 0:     num = num // 10     count = count + 1 print("Чисел:", count) </pre> | <pre> start = 25 end = 50 for num in range(start, end + 1):     if num &gt; 1:         for i in range(2, num):             if (num % i) == 0:                 break         else:             print(num) </pre> |

- а) Программа, которая считает количество чисел в цифре;
- б) Программа, которая выводит все простые числа, из определенного диапазона;
- в) Программа, которая вычисляет сумму всех чисел от 1 до заданного пользователем числа;

Задание № 4. Сопоставьте результаты выполнения программ с предложенными программными решениями.

| Номер | Программное решение |
|-------|---------------------|
|-------|---------------------|

|    |                                                                                                                                                                                                                                                              |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1. | <pre> num = 5 factorial = 1 if num &lt; 0:     print("Для отрицательных чисел, факториал не существует") elif num == 0:     print("Факториал 0 равен 1") else:     for i in range(1, num + 1):         factorial = factorial * i     print(factorial) </pre> |
| 2. | <pre> input_number = 6 print ('Куб числа равен', input_number**3) </pre>                                                                                                                                                                                     |
| 3. | <pre> k = 3 m = 5 n = 9 if n &lt;= k:     t = 2 * m elif n * 2 % k == 0:     t = m * (n * 2 // k) else:     t = m * (1 + (n * 2 // k)) print(t) </pre>                                                                                                       |

Результаты выполнения программ:

- а) 216;
- б) 30;
- в) 120;

Задание № 5. Расположите номера строк предложенного программного кода решения поставленной задачи в правильной последовательности.

Постановка задачи: «Задан год в виде натурального числа. Нужно выяснить, является ли год с данным номером високосным или нет. Если год високосный, то напечатайте "YES", а если нет, то выведите NO. Год является високосным, если он кратен 400 или его номер кратен 4, но не кратен 100».

| Номер строки кода | Строки программного кода        |
|-------------------|---------------------------------|
| 1                 | else:                           |
| 2                 | if a % 4 == 0 and a % 100 != 0: |
| 3                 | a = int(input())                |
| 4                 | print('YES')                    |
| 5                 | print('NO')                     |
| 6                 | print('YES')                    |
| 7                 | elif a % 400 == 0:              |

### Перечень заданий открытого типа

Задание № 1. Укажите, сколько раз выполнится тело цикла с данным заголовком:

```
i=1
while i<=10:
 i=i+1
```

Задание № 2. Что будет выведено на экран монитора в результате выполнения программы:

```
main.py
1 z=5
2 z1=z**3
3 print(z1)
4
```

Задание № 3. Укажите номер строки программы, в которой может возникнуть ошибка? Гарантируется, что на вход программы подаются только целые числа.

```
1 n = int(input())
2 k = int(input())
3 for i in range(n, k):
4 if i % 10 == 0:
5 t += 1
6 else:
7 t = 0
```

Задание № 4. Дан построчный фрагмент программного кода, который принимает три положительных числа и определяет вид треугольника, длины сторон которого равны введенным числам. Дополните программу недостающим оператором в строке кода с номером 1.

| № строки | Программный код                        |
|----------|----------------------------------------|
| 1        |                                        |
| 2        | if a1 == a2 == a3:                     |
| 3        | print('Равносторонний')                |
| 4        | elif a2 == a3 or a1 == a3 or a1 == a2: |
| 5        | print('Равнобедренный')                |
| 6        | else:                                  |
| 7        | print('Разносторонний')                |

Задание № 5. Дан фрагмент программного кода, который принимает целое число x и определяет, принадлежит ли данное число промежутку от -1 до 17. Дополните оператор if в строке с номером 2 необходимым условием.

| № строки | Программный код                      |
|----------|--------------------------------------|
| 1        | <code>a1 = int(input())</code>       |
| 2        | <code>if _____ :</code>              |
| 3        | <code>print('Принадлежит')</code>    |
| 4        | <code>else:</code>                   |
| 5        | <code>print('Не принадлежит')</code> |

### **Формируемая компетенция: ПК 1.1**

#### **Перечень заданий закрытого типа**

Задание №1. На каком этапе разработки программного обеспечения устанавливаются состав и формы представления входной, промежуточной и результирующей информации?

- а) математическое описание;
- б) постановка задачи;
- в) составление программы;
- г) тестирование и отладка программы;
- д) опытная эксплуатация.

Задание № 2. Выберите из предложенных выражений формулировку принципа структурного программирования:

- а) сложная задача разбивается на достаточно простые части, каждая из которых имеет один вход и один выход;
- б) моделирование требуемых атрибутов и взаимодействий сущностей выполняется в виде классов для определения абстрактного представления системы;
- в) создание новых абстракций возможно на основе существующих;
- г) внутреннее состояние и функции объекта скрыты и предоставление доступа осуществляется только через открытый набор функций.

Задание № 3. Выберите из предложенных выражений формулировку принципа объектно-ориентированного программирования:

- а) сложная задача разбивается на достаточно простые части, каждая из которых имеет один вход и один выход;
- б) алгоритм и программа должны составляться поэтапно (по шагам);
- в) создание новых абстракций возможно на основе существующих;
- г) логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

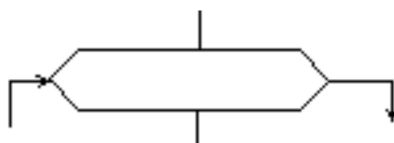
Задание № 4. Как называется алгоритмическая конструкция, обеспечивающая выбор между двумя альтернативами в зависимости от значения входных данных?

- а) разветвляющейся;
- б) циклической;
- в) линейной;
- г) комбинированной.

Задание № 5. Как называется принцип моделирования требуемых атрибутов и взаимодействий сущностей в виде классов для определения основного представления системы?

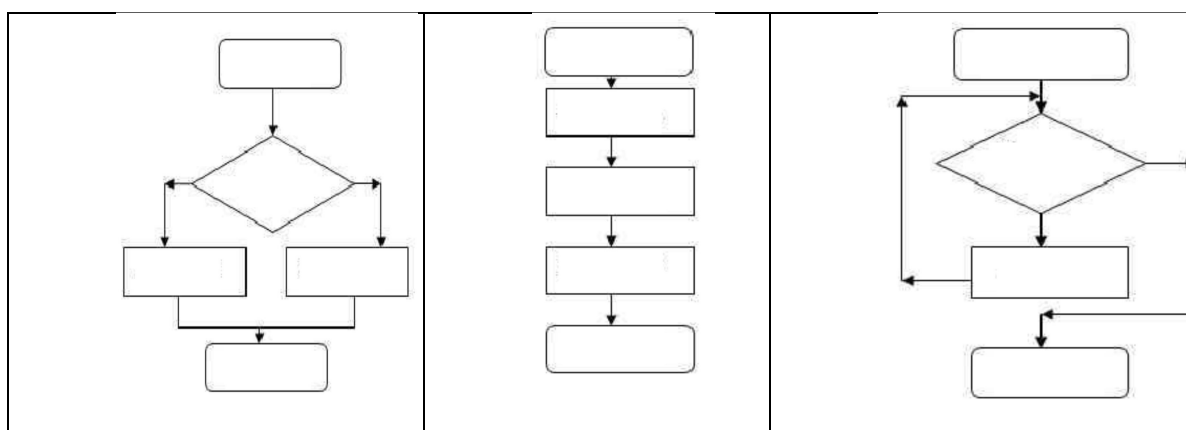
- а) инкапсуляция.
- б) наследование.
- в) абстракция.
- г) полиморфизм.

Задание № 6. Что из нижеперечисленного записывается внутри данного блочного символа при составлении алгоритма решения задачи?



- а) ввод входных данных;
- б) комментарий;
- в) условие;
- г) вычислительное действие;
- д) начальное значение, граничное условие и шаг изменения значения параметра для повторения.

Задание № 7. Сопоставьте названия алгоритмических конструкций с изображениями рисунков?



- а) линейная;
- б) циклическая;
- в) разветвляющаяся.

Задание № 8. Сопоставьте какая алгоритмическая конструкция подходит для каждой из приведенных постановок задач?

| 1                                                 | 2                                                 | 3                                                                                                    |
|---------------------------------------------------|---------------------------------------------------|------------------------------------------------------------------------------------------------------|
| Вычислить произведение и сумму чисел от 10 до 50. | Вычислить значение выражения для $x=8$ и $y=16$ : | Даны два действительных числа. Заменить первое число нулем, если оно меньше второго или равно ему, и |

|  |                                         |                                                  |
|--|-----------------------------------------|--------------------------------------------------|
|  | $z = 1 + \frac{x^2}{2} + \frac{y^3}{4}$ | оставить числа без изменения в противном случае. |
|--|-----------------------------------------|--------------------------------------------------|

- а) линейная;
- б) циклическая;
- в) разветвляющаяся.

Задание № 10. Установите правильную последовательность шагов алгоритма решения квадратного уравнения.

- а) Если ( $D > 0$ ), вычислить  $X1 = (-B + \sqrt{D})/(2*A)$ ,  $X2 = (-B - \sqrt{D})/(2*A)$ , вывести сообщение "Два корня,  $X1 =$ ",  $X1$ , "  $X2 =$ ",  $X2$ .
- б) Вычислить дискриминант  $D = B^2 - 4*A*C$ .
- в) Если ( $D < 0$ ), вывести сообщение "Уравнение действительных решений не имеет".
- г) Если ( $D = 0$ ), вычислить  $X = -B/(2*A)$ , вывести сообщение "Один корень,  $X =$ ",  $X$ .
- д) Получить коэффициенты уравнения -  $A$ ,  $B$ ,  $C$ .

Задание № 11. Установите правильную последовательность этапов решения поставленной задачи на ЭВМ.

- а) построение алгоритма;
- б) формализация задачи;
- в) постановка задачи;
- г) составление программы на языке программирования;
- д) проведение расчетов и анализ полученных результатов;
- е) отладка и тестирование программы.

### Перечень заданий открытого типа

Задание №1. На каком этапе разработки программного обеспечения выбирается форма хранения данных и составляются все необходимые формулы?

Задание №2. Какой принцип объектно-ориентированного программирования формулируется как скрытие внутреннего состояния и функций объекта и предоставление доступа только через открытый набор функций?

Задание №3. Как называется набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата?

Задание №4. Каким блочным символом можно изобразить в блок-схеме следующую строку программного кода: `for i in range(1, num + 1, 2):` ?

Задание №5. Описание структуры алгоритма на естественном, частично формализованном языке, позволяющее выявить основные этапы решения задачи, перед точной его записью на языке программирования называется ...?

Задание №6. Совокупность действий, направленных на устранение ошибок в программах, начиная с момента обнаружения фактов ошибочной работы программы и завершая устранением причин их возникновения называется ...?

**Формируемая компетенция: ПК 1.2**

### **Перечень заданий закрытого типа**

Задание № 1. Какой метод позволяет получить данные от пользователя?

- а) read();
- б) get();
- в) cin();
- г) input().

Задание № 2. Какая функция выводит информацию в консоль (на экран монитора)?

- а) out();
- б) log();
- в) print();
- г) write();

Задание № 3. Какие существуют типы переменных у чисел в Python (выбрать несколько вариантов):

- а) float;
- б) list;
- в) num;
- г) int;
- д) cout;
- е) str.

Задание № 4. Какая результатная информация отобразится на экране после выполнения программы:

Задание № 3. Какие существуют типы переменных у чисел в Python (выбрать несколько вариантов):

- а) float;
- б) list;
- в) num;
- г) int;
- д) cout;
- е) str.

Задание № 4. Какая результатная информация отобразится на экране после выполнения

```
main.py
1 a=5
2 b=a+5
3 a=b*100
4 print(a)
```

- а) 2500;
- б) 1000;
- в) 25;
- г) 100.

Задание № 5. Какая результатная информация отобразится на экране после выполнения программы, если на вход программы подать a=10, b=20?

```
a = int(input())
b = int(input())
if a < b:
 print(a)
else:
 print(b)
```

- а) 10;
- б) 20;
- в) 30;
- г) -10.

Задание № 6. Укажите, значение переменной i после выполнения данного фрагмента программы:

```
i=1
while i<=10:
 i=i+1
```

- а) 9;
- б) 10;
- в) 11;
- г) 2.

Задание № 7. Сопоставьте постановки задач и предложенные программные решения.

| 1                                                                                 | 2                                                                                                                 | 3                                                                                                                                                                                                             |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>n = int(input()) s = 0 for i in range(1,n + 1, 1):     s += i print(s)</pre> | <pre>num = 75869 count = 0 while num != 0:     num = num // 10     count = count + 1 print("Чисел:", count)</pre> | <pre>start = 25 end = 50 for num in range(start, end + 1):     if num &gt; 1:         for i in range(2, num):             if (num % i) == 0:                 break         else:             print(num)</pre> |

- а) Программа, которая считает количество чисел в цифре;
- б) Программа, которая выводит все простые числа, из определенного диапазона;
- в) Программа, которая вычисляет сумму всех чисел от 1 до заданного пользователем числа;

Задание № 8. Сопоставьте результаты выполнения программ с предложенными программными решениями.

| Номер | Программное решение                                                                                                                                                                                                                                        |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1.    | <pre>num = 5 factorial = 1 if num &lt; 0:     print("Для отрицательных чисел, факториал не существует") elif num == 0:     print("Факториал 0 равен 1") else:     for i in range(1, num + 1):         factorial = factorial * i     print(factorial)</pre> |
| 2.    | <pre>input_number = 6 print ('Куб числа равен', input_number**3)</pre>                                                                                                                                                                                     |
| 3.    | <pre>k = 3 m = 5 n = 9 if n &lt;= k:     t = 2 * m elif n * 2 % k == 0:</pre>                                                                                                                                                                              |

|  |                                                                             |
|--|-----------------------------------------------------------------------------|
|  | <pre> t = m * (n * 2 // k) else: t = m * (1 + (n * 2 // k)) print(t) </pre> |
|--|-----------------------------------------------------------------------------|

Результаты выполнения программ:

а) 216;

б) 30;

в) 120;

Задание № 9. Расположите номера строк предложенного программного кода решения поставленной задачи в правильной последовательности.

Постановка задачи: «Задан год в виде натурального числа. Нужно выяснить, является ли год с данным номером високосным или нет. Если год високосный, то напечатайте "YES", а если нет, то выведите NO. Год является високосным, если он кратен 400 или его номер кратен 4, но не кратен 100».

| Номер строки кода | Строки программного кода                                  |
|-------------------|-----------------------------------------------------------|
| 1                 | if (i % n == 0):                                          |
| 2                 | lower = int(input("Введите нижнюю границу диапазона: "))  |
| 3                 | print(i)                                                  |
| 4                 | n = int(input("Введите делитель: "))                      |
| 5                 | upper = int(input("Введите верхнюю границу диапазона: ")) |
| 6                 | for i in range(lower, upper + 1):                         |