



**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**ФГБОУ ВО «Ингушский государственный университет»**

**Гуманитарно-технический колледж**

**СОГЛАСОВАНО**

Заведующий информационно-технического  
отделения

Баркинхоева М.М. \_\_\_\_\_

от « 22 » \_\_\_\_\_ мая \_\_\_\_\_ 2024г.

**УТВЕРЖДАЮ**

Директор ГТК

\_\_\_\_\_ / Дзауров М.А.

от « 24 » \_\_\_\_\_ мая \_\_\_\_\_ 2024г.

## **РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ**

**ОП.04 Основы алгоритмизации и программирования**

для специальности

**09.02.07 Информационные системы и программирование**

по программе базовой подготовки

Магас - 2024г





Рабочая программа разработана на основе Федерального государственного образовательного стандарта среднего профессионального образования по профессиям (специальности) (далее – ФГОС СПО) 09.02.07 Информационные системы и программирование (по отраслям), приказ Министерства образования и науки от 09.12.2016 г. №1547 (Зарегистрировано в Минюсте России 26.12.2016 №44936).

**Организация – разработчик:** ФГБОУ ВО «Ингушский государственный университет» Гуманитарно – технический колледж

**Разработчик:** Горбакова З.А., преподаватель информационно-технического отделения

Рассмотрена на заседании информационно-технического отделения

Протокол № 8 от « 22 » мая 2024 г.

Рассмотрена и одобрена на заседании Методического совета ГТК.

Протокол № 7 от « 23 » мая 20 24 г.



## **СОДЕРЖАНИЕ**

<b>1. ПАСПОРТ РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ</b>	<b>стр. 4</b>
<b>2. СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ</b>	<b>6</b>
<b>3. УСЛОВИЯ РЕАЛИЗАЦИИ УЧЕБНОЙ ДИСЦИПЛИНЫ</b>	<b>15</b>
<b>4. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ</b>	<b>16</b>



# **1 ОБЩАЯ ХАРАКТЕРИСТИКА РАБОЧЕЙ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ «ОП.04 ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ»**

## **1.1. Область применения программы**

Программа учебной дисциплины «ОП.04 Основы алгоритмизации и программирования» является частью ППССЗ в соответствии с ФГОС по специальности среднего профессионального образования по специальностям 09.02.07 Информационные системы и программирование

## **1.2. Место учебной дисциплины в структуре ППССЗ**

Дисциплина «ОП.04 Основы алгоритмизации и программирования» входит в состав общепрофессионального цикла программы среднего профессионального образования – программы подготовки специалистов среднего звена – по специальности 09.02.07 Информационные системы и программирование.

Изучение данного учебного курса является необходимой основой для последующего изучения дисциплин профессиональной подготовки, а также для прохождения учебной и производственной практик, подготовки студентов к государственной итоговой аттестации.

## **1.3. Цели и задачи учебной дисциплины – требования к результатам освоения учебной дисциплины:**

*Целью дисциплины* является изучение и освоение базовых понятий и приемов программирования, применяемых на всех основных этапах разработки программ; изучение методов программирования для овладения знаниями в области технологии программирования; подготовка к осознанному использованию как языков программирования, так и методов программирования.

### ***Задачи:***

- освоить основные методы разработки программного обеспечения;
- приобрести практические навыки программирования для их дальнейшего использования в учебной и профессиональной деятельности.

В результате освоения учебной дисциплины обучающийся должен:

### ***уметь:***

- разрабатывать алгоритмы для конкретных задач.
- использовать программы для графического отображения алгоритмов.
- определять сложность работы алгоритмов.
- работать в среде программирования.
- реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.
- оформлять код программы в соответствии со стандартом кодирования.
- выполнять проверку, отладку кода программы.



**знать:**

- понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции.
- эволюцию языков программирования, их классификацию, понятие системы программирования.
- основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти.
- подпрограммы, составление библиотек подпрограмм
- объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения.

Процесс изучения дисциплины направлен на формирование и развитие следующих компетенций:

- выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам (ОК 01);
- осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности (ОК 02);
- работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами (ОК 04);
- осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста (ОК 05);
- использовать информационные технологии в профессиональной деятельности (ОК 09);
- пользоваться профессиональной документацией на государственном и иностранном языках (ОК 10);
- формировать алгоритмы разработки программных модулей в соответствии с техническим заданием (ПК 1.1);
- разрабатывать программные модули в соответствии с техническим заданием (ПК 1.2);
- выполнять отладку программных модулей с использованием специализированных программных средств (ПК 1.3);
- выполнять тестирование программных модулей (ПК 1.4);
- осуществлять рефакторинг и оптимизацию программного кода (ПК 1.5);
- осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения (ПК 2.4);



– производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования (ПК 2.5).

**1.4. Рекомендуемое количество часов на освоение программы учебной дисциплины:**

максимальной учебной нагрузки обучающегося 166 часов, в том числе: обязательной аудиторной учебной нагрузки обучающегося 152 часа; самостоятельной работы обучающегося 10 часов.

## **2 СТРУКТУРА И СОДЕРЖАНИЕ УЧЕБНОЙ ДИСЦИПЛИНЫ**

### **2.1. Объем учебной дисциплины и виды учебной работы**

<b>Вид учебной работы</b>	<b><i>Объем часов</i></b>
<b>Максимальная учебная нагрузка (всего)</b>	166
<b>Обязательная аудиторная учебная нагрузка (всего)</b>	152
в том числе:	
лекции	76
практические занятия	76
консультации	4
<b>Самостоятельная работа обучающегося (всего)</b>	10
Форма итогового контроля – экзамен	



## 2.2. Тематический план и содержание учебной дисциплины «ОП.04 Основы алгоритмизации и программирования»

Наименование разделов и тем	Содержание учебного материала, лабораторные работы и практические занятия, самостоятельная работа обучающегося, курсовая работа (проект)	Объем в часах	Уровни освоения
1	2	3	4
<b>Раздел 1.</b>	<b>Введение в программирование</b>		
<b>Тема 1.1. Языки программирования</b>	<b>Содержание учебного материала</b>	2	2
	1. Развитие языков программирования.		
	2. Обзор языков программирования. Области применения языков программирования. Стандарты языков программирования. Среда проектирования. Компиляторы и интерпретаторы.		
	3. Жизненный цикл программы. Программа. Программный продукт и его характеристики.		
	4. Основные этапы решения задач на компьютере.		
	<b>Практическая работа №1.</b> Составление блок-схем алгоритмов.	6	2
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования Подготовить сообщение «Применение алгоритмов на практике»	1	3
<b>Тема 1.2. Типы данных</b>	<b>Содержание учебного материала</b>	2	2
	1. Типы данных. Простые типы данных. Производные типы данных. Структурированные типы данных.		
	<b>Практическая работа №2.</b> Составление словесных алгоритмов.	6	2
<b>Раздел 2.</b>	<b>Содержание учебного материала</b>		
<b>Тема 2.1. Операторы языка программирования</b>	1. Операции и выражения. Правила формирования и вычисления выражений. Структура программы. Ввод и вывод данных. Оператор присваивания. Составной оператор.	4	2
	2. Условный оператор. Оператор выбора.		
	3. Цикл с постусловием. Цикл с предусловием. Цикл с параметром. Вложенные циклы.		
	4. Массивы. Двумерные массивы. Строки. Стандартные процедуры и функции для работы со строками.		
	5. Структурированный тип данных – множество. Операции над множествами.		



	6. Комбинированный тип данных – запись. Файлы последовательного доступа. Файлы прямого доступа		
	<b>Практическая работа № 3.</b> Знакомство с Паскаль. <b>Практическая работа № 4.</b> Составление программ линейной структуры. <b>Практическая работа №5.</b> Составление программ разветвляющейся структуры. <b>Практическая работа №6.</b> Составление программ циклической структуры. <b>Практическая работа №7.</b> Составление программ усложненной структуры. <b>Практическая работа №8.</b> Обработка одномерных массивов. <b>Практическая работа №9.</b> Обработка двумерных массивов. <b>Практическая работа №10.</b> Работа со строковыми переменными. <b>Практическая работа №11.</b> Работа с данными типа множество.	18	2
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования; <ul style="list-style-type: none"> <li>– Подготовить доклад: Изучение организация ввода-вывода данных;</li> <li>– Подготовить сообщение: Одномерные массивы;</li> <li>– Подготовить сообщение: Двумерные массивы;</li> <li>– Составление словаря терминов, используемых в системе программирования;</li> <li>– Составление словаря терминов, используемых в системе программирования;</li> <li>– Составление блок-схемы программ по практической работе «Использование стандартных функций для работы со строками»;</li> <li>– Составление словаря терминов, используемых в системе программирования;</li> <li>– Подготовить сообщение: Интегрированная среда программирования Pascal;</li> <li>– Подготовить сообщение: Операторы языка Pascal;</li> <li>– Подготовить презентацию: Одномерные массивы. Двумерные массивы.</li> </ul>	1	3
<b>Раздел 3.</b>	<b>Содержание учебного материала</b>		
<b>Тема 3.1. Процедуры и функции</b>	1. Общие сведения о подпрограммах. Определение и вызов подпрограмм. Область видимости и время жизни переменной. Механизм передачи параметров. Организация функций.	2	2
	2. Рекурсия. Программирование рекурсивных алгоритмов.		
	<b>Практическая работа №12.</b> Организация процедур. <b>Практическая работа №13.</b> Использование процедур. <b>Практическая работа №14.</b> Организация функций. <b>Практическая работа №15.</b> Использование функций. <b>Практическая работа №16.</b> Использование стандартных функций и процедур для	12	2



	работы со строками. <b>Практическая работа №17.</b> Использование стандартных функций для работы с массивами.		
<b>Тема 3.2.</b> <b>Структуризация в программировании</b>	<b>Содержание учебного материала</b>		
	1. Основы структурного программирования. Методы структурного программирования.	2	2
	<b>Практическая работа №18.</b> Работа с файлом последовательного доступа. <b>Практическая работа №19.</b> Работа с файлом произвольного доступа. <b>Практическая работа №20.</b> Использование стандартных процедур и функций для работы с файлами.	6	2
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования Подготовить сообщение: Ознакомление со стандартными библиотеками подпрограмм; Выполнение задач по теме «Модули» Составить схему вызова библиотек	2	3
<b>Тема 3.3.</b> <b>Модульное программирование</b>	<b>Содержание учебного материала</b>		
	1. Модульное программирование. Понятие модуля. Структура модуля. Компиляция и компоновка программы.	8	2
	2. Стандартные модули.		
	<b>Практическая работа №21.</b> Программирование модуля. <b>Практическая работа №22.</b> Создание библиотеки подпрограмм.	4	2
<b>Раздел 4</b>	<b>Основные конструкции языков программирования</b>		
<b>Тема 4.1</b> <b>Указатели.</b>	<b>Содержание учебного материала</b>		
	1. Указатели. Описание указателей. Основные понятия и применение динамически распределяемой памяти. Создание и удаление динамических переменных.	7	2
	2. Структуры данных на основе указателей.		
	3. Задача о стеке.		
	<b>Практическая работа №23.</b> Использование библиотеки подпрограмм.	2	2
<b>Раздел 5</b>	<b>Содержание учебного материала</b>		
<b>Тема 5.1 Основные</b>	1. История развития ООП. Базовые понятия ООП: объект, его свойства и методы, класс, интерфейс.	2	2
	2. Основные принципы ООП: инкапсуляция, наследование, полиморфизм.		



<b>принципы объектно- ориентированного программирования (ООП)</b>	3. Классы объектов. Компоненты и их свойства.		
	4. Событийно-управляемая модель программирования. Компонентно-ориентированный подход.		
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования	1	3
<b>Тема 5.2 Интегрированная среда разработчика.</b>	<b>Содержание учебного материала</b>		
	1. Требования к аппаратным и программным средствам интегрированной среды разработчика.		
	2. Интерфейс среды разработчика: характеристика, основные окна, инструменты, объекты. Форма и размещение на ней управляющих элементов.		
	3. Панель компонентов и их свойства. Окно кода проекта.	7	2
	4. Состав и характеристика проекта. Выполнение проекта. Настройка среды и параметров проекта.		
	5. Панель компонентов и их свойства. Окно кода проекта. Состав и характеристика проекта. Выполнение проекта. Настройка среды и параметров проекта.		
	6. Настройка среды и параметров проекта.		
	<b>Практическая работа №24.</b> Изучение интегрированной среды разработчика. <b>Практическая работа №25.</b> Создание простого проекта.	4	2
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования Составить памятку: Этапы разработки приложения;	1	3
<b>Тема 5.3. Визуальное событийно- управляемое программирование</b>	<b>Содержание учебного материала</b>		
	1. Основные компоненты (элементы управления) интегрированной среды разработки, их состав и назначение.		
	2. Дополнительные элементы управления. Свойства компонентов. Виды свойств. Синтаксис определения свойств. Назначения свойств и их влияние на результат. Управление объектом через свойства.	10	2
	3. События компонентов (элементов управления), их сущность и назначение. Создание процедур на основе событий.		
	<b>Практическая работа № 26</b> Создание проекта с использованием кнопочных компонентов. <b>Практическая работа № 27</b> Создание проекта с использованием компонентов для работы с текстом. <b>Практическая работа № 28</b> Создание проекта с использованием компонентов	6	2



	ввода и отображения чисел, дат и времени. <b>Лабораторная работа № 29</b> Создание проекта с использованием компонентов стандартных диалогов и системы меню.		
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования Подготовить реферат: Визуальное событийно-управляемое программирование Составить памятку: Свойства основных компонентов интегрированной среды обработки	1	3
<b>Тема 5.4</b> <b>Разработка</b> <b>оконного</b> <b>приложения</b>	<b>Содержание учебного материала</b>	10	2
	1. Разработка функционального интерфейса приложения. Создание интерфейса приложения.		
	2. Разработка функциональной схемы работы приложения.		
	3. Разработка игрового приложения.	4	2
	<b>Практическая работа №30.</b> Разработка оконного приложения. <b>Практическая работа №31.</b> Разработка оконного приложения с несколькими формами.		
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования Составить памятку: Разработка оконного приложения Подготовка к экзамену	1	3
<b>Тема 5.5 Этапы</b> <b>разработки</b> <b>приложений</b>	<b>Содержание учебного материала</b>	10	2
	1. Разработка приложения.		
	2. Проектирование объектно-ориентированного приложения.		
	3. Создание интерфейса пользователя.		
	4. Тестирование, отладка приложения.	2	2
	<b>Практическая работа №32</b> Изучение этапов разработки		
<b>Тема 5.6 Иерархия</b> <b>классов.</b>	<b>Содержание учебного материала</b>	10	2
	1. Классы ООП: виды, назначение, свойства, методы, события.		
	2. Перегрузка методов.		
	3. Тестирование и отладка приложения.		
	4. Решение задач	6	2
	<b>Практическая работа №33.</b> Объявление класса, создание экземпляров класса. <b>Практическая работа №34.</b> Создание наследованного класса.		



	<b>Практическая работа №35</b> Перегрузка методов.		
	<b>Самостоятельная работа обучающихся:</b> Составление словаря терминов, используемых в системе программирования Составить памятку: Интегрированная среда разработчика; Составить схему: Классы объектно-ориентированного языка программирования Составить сообщение: Иерархия классов	2	3
	<b>Консультации</b>	<b>4</b>	
<b>Всего:</b>		<b>166</b>	

Для характеристики уровня освоения учебного материала используются следующие обозначения:

1. – ознакомительный (узнавание ранее изученных объектов, свойств);
2. – репродуктивный (выполнение деятельности по образцу, инструкции или под руководством)
3. – продуктивный (планирование и самостоятельное выполнение деятельности, решение проблемных задач)



### **3 УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ УЧЕБНОЙ ДИСЦИПЛИНЫ**

#### **3.1. Материально-техническое обеспечение**

Для реализации программы учебной дисциплины предусмотрена лаборатория программирования и баз данных, оснащенная необходимым оборудованием:

- автоматизированные рабочие места для обучающихся (процессор Core i7, оперативная память 8 Гб, дискретная видеокарта 2GB ОЗУ, монитор 24”) (12 шт.);
- автоматизированное рабочее место преподавателя (процессор Core i7, оперативная память 8 Гб; дискретная видеокарта 2GB ОЗУ монитор 24”) (1 шт.);
- сервер в серверной (8-х ядерный процессор 3 ГГц, оперативная память 16 Гб, жесткие диски общим объемом 1 Тб) (1 шт.);
- выделен виртуальный сервер из общей фермы серверов (1 шт.);
- проектор мультимедийный (1 шт.);
- интерактивная доска (1 шт.);
- видеокамера Hikvision (1 шт.);
- маркерная доска (1 шт.).

#### **3.2. Информационное обеспечение обучения**

Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы

##### *Основная литература*

1. Семакин, И. Г. Основы алгоритмизации и программирования [Текст] : учеб. для студ-ентов учреждений сред. проф. образования / И. Г. Семакин, А. П. Шестаков. – 2-е изд., стер. – М. : Академия, 2018. – 301 с.

##### *Дополнительная литература*

1. Семакин, И. Г. Основы алгоритмизации и программирования. Практикум [Текст] : учеб. пособие для студентов учреждений сред. проф. образования / И. Г. Семакин, А. П. Шестаков. – 5-е изд., стер. – М. : Академия, 2017. – 141 с.
2. Лубашева, Т. В. Основы алгоритмизации и программирования [Электронный ресурс] : учеб. пособие / Т. В. Лубашева, Б. А. Железко. – Минск : РИПО, 2016. – 378 с. – Режим доступа : <http://biblioclub.ru/index.php?page=book&id=463632>



## 4. КОНТРОЛЬ И ОЦЕНКА РЕЗУЛЬТАТОВ ОСВОЕНИЯ УЧЕБНОЙ ДИСЦИПЛИНЫ

**Контроль и оценка** результатов освоения учебной дисциплины осуществляется преподавателем в процессе проведения практических занятий и лабораторных работ, тестирования, а также выполнения курсового проектирования.

Результаты обучения	Критерии оценки	Формы и методы оценки
<p><i>Перечень умений, осваиваемых в рамках дисциплины:</i></p> <ul style="list-style-type: none"> <li>– Разрабатывать алгоритмы для конкретных задач.</li> <li>– Использовать программы для графического отображения алгоритмов.</li> <li>– Определять сложность работы алгоритмов.</li> <li>– Работать в среде программирования.</li> <li>– Реализовывать построенные алгоритмы в виде программ на конкретном языке программирования.</li> <li>– Оформлять код программы в соответствии со стандартом кодирования.</li> <li>– Выполнять проверку, отладку кода программы.</li> </ul>	<p>«Отлично» - теоретическое содержание курса освоено полностью, без пробелов, умения сформированы, все предусмотренные программой учебные задания выполнены, качество их выполнения оценено высоко.</p> <p>«Хорошо» - теоретическое содержание курса освоено полностью, без пробелов, некоторые умения сформированы недостаточно, все предусмотренные программой учебные задания выполнены, некоторые виды заданий выполнены с ошибками.</p>	<ul style="list-style-type: none"> <li>– Тестирование на знание терминологии по теме</li> <li>– Контрольная работа</li> <li>– Самостоятельная работа</li> <li>– Защита реферата</li> <li>– Защита курсовой работы (проекта)</li> <li>– Наблюдение за выполнением практического задания. (деятельностью студента)</li> <li>– Оценка выполнения практического задания (работы)</li> <li>– Подготовка и выступление с докладом, сообщением, презентацией</li> </ul>
<p><i>Перечень знаний, осваиваемых в рамках дисциплины:</i></p> <ul style="list-style-type: none"> <li>– Понятие алгоритмизации, свойства алгоритмов, общие принципы построения алгоритмов, основные алгоритмические конструкции.</li> <li>– Эволюцию языков программирования, их классификацию, понятие системы программирования.</li> <li>– Основные элементы языка, структуру программы, операторы и операции, управляющие структуры, структуры данных, файлы, классы памяти.</li> <li>– Подпрограммы, составление библиотек подпрограмм</li> </ul>	<p>«Удовлетворительно» - теоретическое содержание курса освоено частично, но пробелы не носят существенного характера, необходимые умения работы с освоенным материалом в основном сформированы, большинство предусмотренных программой обучения учебных заданий выполнено, некоторые из выполненных заданий содержат ошибки.</p> <p>«Неудовлетворительно» -</p>	



<p>– Объектно-ориентированную модель программирования, основные принципы объектно-ориентированного программирования на примере алгоритмического языка: понятие классов и объектов, их свойств и методов, инкапсуляция и полиморфизма, наследования и переопределения.</p>	<p>теоретическое содержание курса не освоено, необходимые умения не сформированы, выполненные учебные задания содержат грубые ошибки.</p>	
---	---	--



## **ФОНД ОЦЕНОЧНЫХ СРЕДСТВ**

### **Перечень формируемых компетенций**

Процесс изучения дисциплины направлен на формирование и развитие следующих компетенций:

- выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам (ОК 01);
- осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности (ОК 02);
- работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами (ОК 04);
- осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста (ОК 05);
- использовать информационные технологии в профессиональной деятельности (ОК 09);
- пользоваться профессиональной документацией на государственном и иностранном языках (ОК 10);
- формировать алгоритмы разработки программных модулей в соответствии с техническим заданием (ПК 1.1);
- разрабатывать программные модули в соответствии с техническим заданием (ПК 1.2);
- выполнять отладку программных модулей с использованием специализированных программных средств (ПК 1.3);
- выполнять тестирование программных модулей (ПК 1.4);
- осуществлять рефакторинг и оптимизацию программного кода (ПК 1.5);
- осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения (ПК 2.4);
- производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования (ПК 2.5).



**3.1. Типовые контрольные задания или иные материалы,  
необходимые для оценки знаний, умений, навыков и (или) опыта деятельности,  
характеризующих этапы формирования компетенций при изучении учебной  
дисциплины в процессе освоения образовательной программы**

**Типовой тест промежуточной аттестации**

**Учащиеся должны знать:**

- определение алгоритма и его основные свойства;
- способы записи алгоритмов;
- назначение языков программирования, систем программирования;
- структуру программы;
- операторы ввода, вывода, присваивания;
- алгоритмические структуры;
- типы переменных;
- функции в языках объектно-ориентированного и процедурного программирования.

**Критерии оценивания:**

**Оценка «3»** - за 7-10 правильных ответов;

**Оценка «4»** - за 11-13 правильных ответов;

**Оценка «5»** - за 14-15 правильных ответов;

**Ответы к тесту:**

**1 вариант**

вопроса

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Ответ

в

а

а

г

б

б

в

а

г

б

в



в  
б  
а  
г

## 2 вариант

вопроса

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15

Ответ

в  
б  
б  
г  
а  
б  
г  
в  
в  
б  
б  
в  
б  
в  
д

1. Алгоритм — это:

- а) правила выполнения определенных действий;
- б) ориентированный граф, указывающий порядок исполнения некоторого набора команд;
- в) понятное и точное предписание исполнителю совершить последовательность действий, направленных на достижение поставленных целей;
- г) набор команд для компьютера.

2. Алгоритм называется циклическим, если:

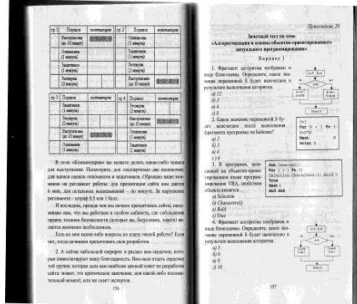
- а) он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
- б) ход его выполнения зависит от истинности тех или иных условий;
- в) его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий;



г) он представим в табличной форме.

3. Свойством алгоритма является:

- а) результативность;
- б) цикличность;
- в) возможность изменения последовательности выполнения команд;
- г) возможность выполнения алгоритма в обратном порядке.



4. Фрагмент алгоритма изображен в виде блок-схемы.

Определите, какое значение переменной S будет напечатано в результате выполнения алгоритма.

- а) 12
- б) 3
- в) 4
- г) 8

5. В программе, записанной на объектно-ориентированном языке программирования VB, свойством объекта



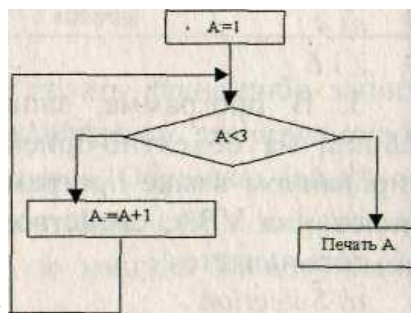
является

- а) Selection
- б) Characters(i)
- в) Bold
- г) True

6. Какое значение переменной S будет напечатано после выполнения фрагмента программы наBasic?

- а) 5
- б) 7
- в) 9
- г) 12

7. Фрагмент алгоритма изображен в виде блок-схемы.



Определите, какое значение переменной A будет напечатано в результате выполнения алгоритма.

- а) 1
- б) 2
- в) 3
- г) 4

8. В программе, записанной на объектно-ориентированном языке программирования VB, объектом является...

- а) Form1



- б) Print
- в) Command1\_Click()
- г) Int

9. Выявление ошибок и их устранение называется ...

- а) отладкой задачи; в) отладкой алгоритма
- б) отладкой исполнителя; г) отладкой программы?

10. Человек, робот, автомат, устройство, компьютер, который выполняет чьи-то команды - это ...

- а) помощник в) программа
- б) исполнитель г) раб

11. Повторяющийся блок действий (команд) называется ...

- а) повтором; в) телом цикла;
- б) циклом; г) командой повторения

12. Совокупность всех команд, которые может выполнить конкретный исполнитель, - это ...

- а) система программ; в) система команд;
- б) система алгоритмов; г) система задач

13. Команда, у которой действия выполняются после проверки условия, называется ...

- а) командой цикла; в) командой ветвления;
- б) простой командой; г) процедурой

14. Свойство алгоритма, заключающееся в том, что алгоритм должен состоять из конкретных действий, следующих в определенном порядке, называется

- а) дискретность;
- б) детерминированность;
- в) конечность;
- г) массовость;
- д) результативность.

15. Свойство алгоритма, заключающееся в том, что один и тот же алгоритм можно использовать с разными исходными данными, называется

- а) дискретность;
- б) детерминированность;
- в) конечность;
- г) массовость;
- д) результативность.

1. Алгоритм называется линейным, если:

- а) он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;
- б) ход его выполнения зависит от истинности тех или иных условий;



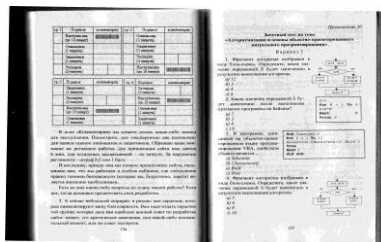
- в) его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий;  
г) он представим в табличной форме.

2. Алгоритм включает в себя ветвление, если:

- а) он составлен так, что его выполнение предполагает многократное повторение одних и тех же действий;  
б) ход его выполнения зависит от истинности тех или иных условий;  
в) его команды выполняются в порядке их естественного следования друг за другом независимо от каких-либо условий;  
г) он представим в табличной форме.

3. Вместо многоточия вставьте подходящий ответ для следующего утверждения: «От любого исполнителя не требуется...»:

- а) соблюдать последовательность действий;  
б) понимать смысл алгоритма;  
в) формально выполнять команды алгоритма;  
г) умение точно выполнять команды.

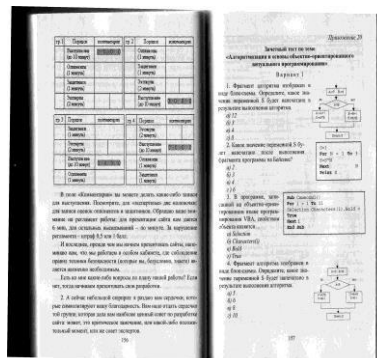


4. Какое значение переменной S будет напечатано после выполнения фрагмента программы на Basic?

- а) 2  
б) 3  
в) 4  
г) 6

5. Фрагмент алгоритма значение выполнения алгоритма.

- а) 5  
б) 6  
в) 8  
г) 10



изображен в виде блок-схемы. Определите, какое значение переменной S будет напечатано в результате

6. В программе, записанной на объектно-ориентированном языке программирования VB методом объекта является...

- а) Documents ( )  
б) Open  
в) File Name  
г) "C:\Проба.doc"

7. Какое значение переменной S будет напечатано после выполнения фрагмента программы на Basic?

- а) 2  
б) 3  
в) 5  
г) 6



8. Запись алгоритма на языке конкретного исполнителя – это ...  
а) алгоритм; в) команда;  
б) программа; г) исполнитель?
9. Отдельное указание исполнителю – это ...  
а) программа; в) команда;  
б) алгоритм; г) приказ?
10. Форма организации действий, при которой один и тот же блок выполняется несколько раз, называется ...  
а) следованием; в) ветвлением;  
б) циклом; г) алгоритмом?
11. Составная команда, в которой одни и те же действия (команды) повторяются несколько раз, называется ...  
а) командой присваивания; в) командой повторения;  
б) вспомогательной программой; г) командой ветвления?
12. Вспомогательная команда – это ...  
а) цикл; в) процедура;  
б) ветвление; г) следование?
13. Графический способ описания алгоритма – это ...  
а) программа; в) алгоритм;  
б) блок-схема; г) словесно-пошаговая запись?
14. Свойство алгоритма, заключающееся в том, что каждое действие и алгоритм в целом должны иметь возможность завершения, называется  
а) дискретность;  
б) детерминированность;  
в) конечность;  
г) массовость;  
д) результативность.
15. Свойство алгоритма, заключающееся в отсутствии ошибок, алгоритм должен приводить к правильному результату для всех допустимых входных значениях, называется  
а) дискретность;  
б) детерминированность;  
в) конечность;  
г) массовость;  
д) результативность.

#### **Экзаменационные вопросы**

1. Переменные. Типы данных в C++.
2. Структура программы. Команда присваивания в C++.
3. Ввод-вывод данных. Формат выводимых данных.
4. Ввод-вывод данных. Стандартные потоки ввода и вывода. Примеры.
5. Алгоритм линейной структуры в C++.
6. Структура IF, классификация в C++. Примеры.
7. Структура switch(выбор) и ее программирование в C++. Примеры.
8. Алгоритмы циклической итерационной структуры. Оператор цикла While в C++. Примеры использования.



9. Алгоритмы циклической итерационной структуры. Оператор цикла do... while в C++.  
Примеры использования.
10. Алгоритмы циклической итерационной структуры. Оператор цикла For в C++. Примеры использования.
11. Операторы break и continue в C++. Примеры использования.
12. Одномерные массивы в C++. Задание массивам первоначальных значений.
13. Операции над массивами и их совместимость. Ввод-вывод массивов в C++.
14. Случайные числа в языке программирования C++.
15. Понятие подпрограммы в C++. Описание подпрограммы.
16. Формальные и фактические параметры в C++.
17. Понятие о локальных и глобальных переменных в C++.
18. Основные математические функции в C++. Примеры.
19. Определение алгоритма. Свойства алгоритма. Формы записи алгоритмов. Примеры.
20. Запись алгоритмов блок-схемами. Основные элементы блок-схем.
21. Алгоритмы с ветвлением. Пример алгоритма.
22. Алгоритм цикла с предусловием. Пример алгоритма.
23. Алгоритм цикла с постусловием. Пример алгоритма.
24. Алгоритм цикла с управляющей переменной. Пример алгоритма.
25. Переменные. Типы данных в Python.
26. Структура программы. Команда присваивания в Python.
27. Ввод-вывод данных. Формат выводимых данных в Python.
28. Алгоритм линейной структуры в Python.
29. Структура IF, классификация в Python. Примеры.
30. Алгоритмы циклической итерационной структуры. Оператор цикла While в Python. Примеры использования.
31. Строки как последовательности символов в Python. Примеры использования.
32. Алгоритмы циклической итерационной структуры. Оператор цикла обхода For в Python. Примеры использования.
33. Оператор break в Python. Примеры использования.
34. Оператор continue в Python. Примеры использования.
35. Списки – изменяемые последовательности в Python. Примеры использования.
36. Словари в Python. Примеры использования.
37. Понятие подпрограммы в Python. Описание подпрограммы.
38. Формальные и фактические параметры в Python.
39. Понятие о локальных и глобальных переменных в Python.
40. Функции в Python.

### **Практическая работа №1. Составление блок-схем алгоритмов.**

**Цель работы:** сформировать умения по составлению алгоритма программы; научить записывать его с помощью блок-схем.

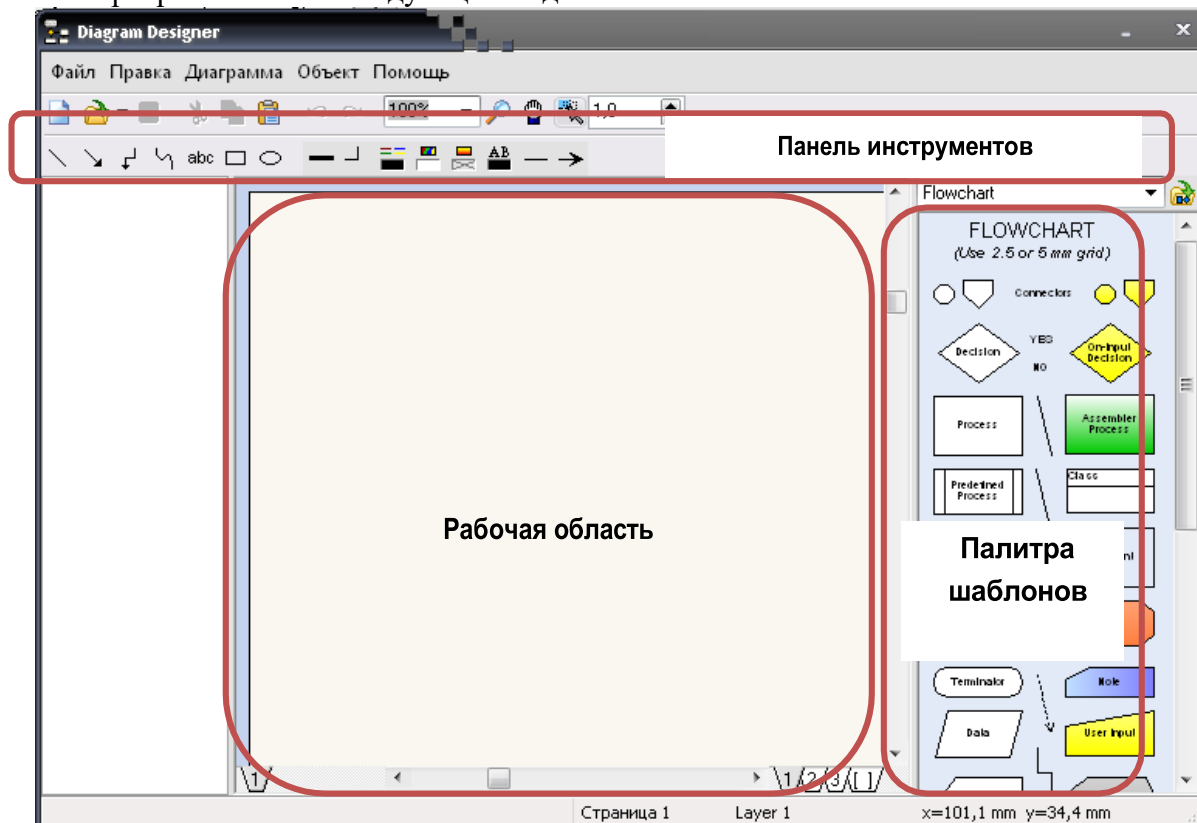
**Оборудование, технические и программные средства:** персональный компьютер, приложение DiagramDesigner.



## Задание 1. Создание простейшей блок-схемы

### Методические указания по выполнению задания:

1. Запустите приложение **DiagramDesigner**. **DiagramDesigner** является приложением для создания блок-схем. В целом приложение способно соединять любые «блоки» поэтому в качестве блоков может выступать даже элементы электрической принципиальной схемы. Элементы-блоки можно составлять пользователю. Так же заявлены возможность построения графиков, слайд-шоу и встроенный калькулятор.
2. Окно программы имеет следующий вид:




3. Выберем шаблон, который был специально создан для данной практической работы, для этого скопируйте файл **simple** в папку своей группы. Далее в окне программы щелкните правой кнопкой мыши в палитре шаблонов и в контекстном меню выполните команду **Загрузите палитру шаблонов** и выберите файл **simple**.
4. Разместите в рабочей области прямоугольный блок, являющийся блоком Действия. Для этого нажмите левую кнопку мыши на блоке Действия и перетащите на рабочую область и отпустите левую кнопку мыши. Теперь вы умеете добавлять блоки на блок-схему – в основном вы будете добавлять блоки перемещением с палитры шаблонов и расположении их в рабочей области.
5. Далее необходимо уточнить, что же за действие выполняется, ведь иначе блок не имеет смысла, для этого необходимо выполнить двойной щелчок левой кнопкой мыши на блоке, в



открывшемся окне **Редактировать текст** введите текст **Поставить чайник на газ** и нажмите кнопку **Хорошо**. В результате текст внутри блока изменится на то, что мы ввели в окне редактора.

6. Добавьте блок действия и измените его описание на **Достать чашку**. После, добавьте ещё один блок действия с описанием **Засыпать заварку**. И наконец, добавьте последний блок действия **Добавить сахар по вкусу**. Блок-схема примет следующий вид:



7. Блоки раскиданы в рабочей области произвольно, и это мало напоминает блок-схему. Конечно, можно расставить их более-менее в «столбец», но это тяжело и нудно. Поэтому необходимо выполнить соединение блоков. Для этого щелкните левой кнопкой мыши на элементе **Линия**  на панели инструментов. Нажмите левую кнопку мыши на центре нижней рамки блока **Поставить чайник на газ**, там изображено красное перекрестие. Далее наведите на центр верхней рамки блока **Достать чашку**, при наведении крестик меняет цвет с красного на зеленый, отпустите левую кнопку мыши. Теперь эти два блока соединены. Чтобы убедиться, что блоки были действительно соединены, выделите то пространство где хотите сделать такую проверку. В этой области все соединения будут подсвечены «зеленым», т.е. крестики будут вместо красных – зеленые. Так соединяются не только блоки действий, а все блоки на блок-схеме. Важно лишь, что можно соединять их только за те места (обозначены красными крестами) которые предусмотрены блоком.
8. Выполним выравнивание блоков на блок-схеме. Для этого выделите блок **Достать чашку** левой кнопкой мыши и не отпуская её передвигайте блок, пока стрелка соединяющая блоки не выровняется.
9. Соедините и выровняйте все оставшиеся блоки.



10. Добавьте на блок-схему блок нового типа – блок **Условия**. Для этого нажмите левую кнопку мыши на блоке **Условия** и перетащите его на рабочую область. Как видите, добавление блока **Условия** ничем не отличается от добавления блока Действия.



11. Соедините блок действия **Добавить сахар по вкусу** и новоиспеченный блок условия.
12. Измените описание блока условия со стандартного **Условный(ые) оператор(ы)** на **Чайник не кипит**.
13. Условие **Чайник не кипит** может быть ДА или НЕТ. Все, что внутри блока условия преобразуется к ДА и НЕТ, чтобы получить разветвление пути программы. Принято путь выполнения ДА изображать под блоком условия, а путь выполнения НЕТ справа от него, но не на том же уровне, а ниже. Чтобы провести связи от блока условия создадим те блоки, что будут идти в пути «ДА» и в пути «НЕТ». Добавьте блоки действий **Ждать минуту** - под блоком условия, а блок **Залить чай** справа от него. Вот что должно получиться:

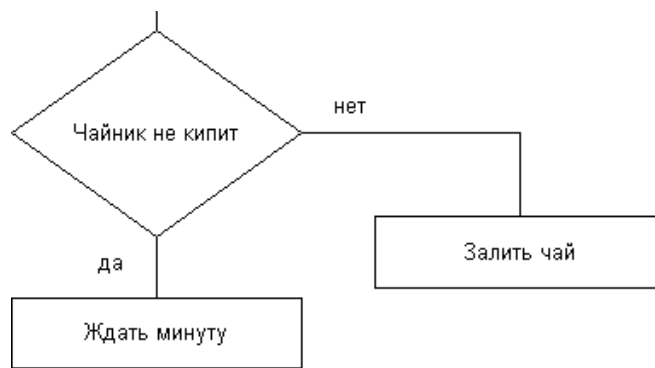


14. Для пути ДА легко будет соединить обычной линией, в то время как для соединения блока действия **Залить чай** с блоком условия **Чайник не кипит** необходимо либо две линии, либо коннектор. В блок-схемах не допускаются линии «наискосок», поэтому одной линией не обойтись. Коннектор есть на палитре шаблонов, и более того – для лучшего понимания он как раз и соединяет блок условия с путем НЕТ.

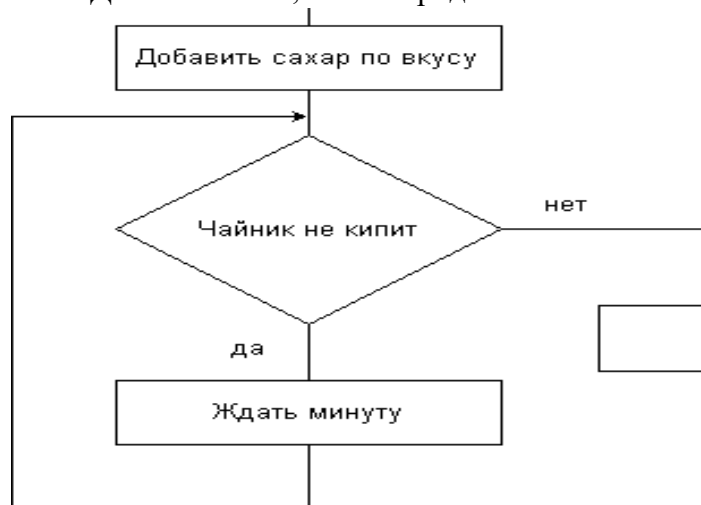


15. В результате мы имеем разветвление работы алгоритма (так же будет и с блок-схемами программ), но все же стоит указать где ДА и НЕТ тем более вначале это не совсем очевидно. Сделайте это так как показано на рисунке.



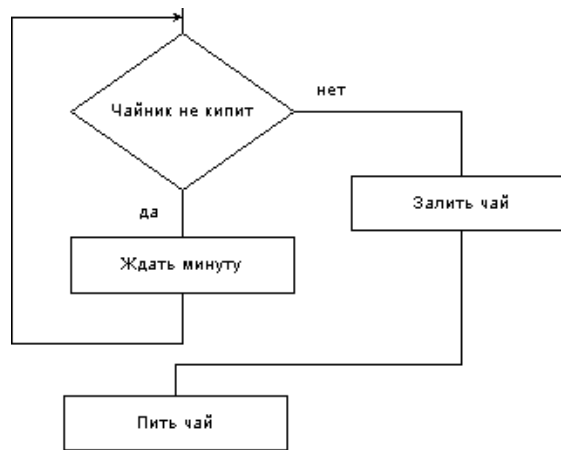


16. Теперь прекрасно видно, что когда происходит. Осталось только подумать что будет после действия **Ждать минуту** и после действия **Залить чай** и добавить это. После **Ждать минуту** очевидно, нужно снова проверить чайник. Т.е. мы возвращаемся в место перед блоком условия **Чайник не кипит**. Сделайте небольшую линию вниз от блока **Ждать минуту**, к ней присоедините ещё одну линию, идущую на несколько сантиметров влево, потом – линия вверх, на уровень середины линии между блоками **Добавить сахар по вкусу** и **Чайник не кипит**. Должно выйти, что-то вроде этого:



17. То, что мы сейчас организовали, в программировании называется цикл. На самом деле есть специальные блоки для циклов, но нагляднее изобразить их так. Важно же в блок-схеме, то, что мы можем создать принцип работы нашей программы, и лучше осознать как сделать её.
18. Не забывая и про блок **Залить чай** продолжим его путь. Для этого стрелочками образуем дорогу вниз, а потом влево, так чтобы оказаться снова на «осевой линии» блок-схемы. Таковы правила, они делают блок-схемы более наглядными. После этого добавим блок **Пить чай** и, соединив его с последней линией, закончим блок-схему:



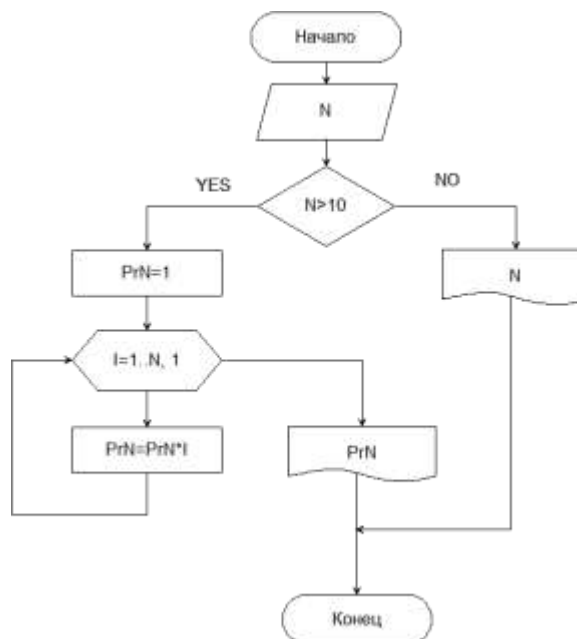


19. Сохраните блок-схему в папке своей группы под именем **схема1.ddd**

## Задание 2. Создание блок-схемы алгоритма программы

### Методические указания по выполнению задания:

- Создайте блок-схему алгоритма программы, которая запрашивает с клавиатуры целое число  $N$  и если это число больше 10, то вычисляет и выводит на экран произведение всех целых чисел от 1 до  $N$ . Составим алгоритм решения данной задачи на естественном языке:
  - Ввод исходных данных  $N$ .
  - Проверка условия  $N > 10$ .
  - В случае выполнения условия вычисляется произведение всех целых чисел от 1 до  $N$ .
  - Вывод на экран полученного произведения и завершение выполнения программы.
  - В противном случае вывод на экран значения  $N$  и завершения программы.
- Запишите данный алгоритм на языке блок-схем. Для записи алгоритма воспользуйтесь программным средством **DiagramDesigner**.
- Создайте новую блок-схему. Затем в левой части рабочего окна выберите соответствующий шаблон и нанесите его на рабочую область. Для задания надписей на блоках, выполните двойной щелчок по соответствующему блоку.
- Оформите блок схему в соответствии с рисунком:



- Сохраните построенную блок-схему в папке своей группы под именем **схема2.ddd**

### Индивидуальные задания:

Каждый студент должен выполнить все задания.



- Напишите блок-схему программы, которая запрашивает у пользователя номер одного из весенних месяцев, и выводит количество дней в этом месяце. Программа должна проверять является ли введенный месяц весенним.
- Напишите блок-схему программы, которая запрашивает с клавиатуры число X. Если X меньше 10, то вычисляет и выводит на экран квадрат числа X, а если больше или равно 10, то вводит новое число Y, а затем вычисляет и выводит на экран значение суммы X и Y.

## Практическая работа №2

### «Состав среды программирования. Состав окна, меню программы. Ввод текста программы в окне редактора»

**Цель работы:** сформировать практические навыки работы с системой PascalABC.Net; научиться создавать, вводить в компьютер, выполнять и исправлять простейшие программы на языке PascalABC.Net.

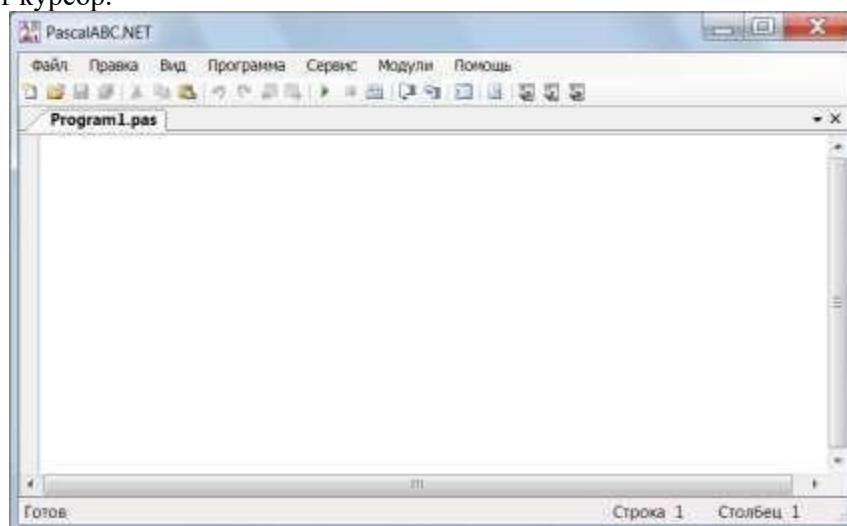
**Оборудование, технические и программные средства:** персональный компьютер, система программирования PascalABC.NET.

#### Задание 1.

Познакомьтесь со способами загрузки среды программирования **PascalABC.Net**, её интерфейсом. Изучите назначение команд меню системы программирования **PascalABC.Net**.

#### Методические указания по выполнению задания:



1. Система **PascalABC** основана на языке **Delphi Pascal** и призвана осуществить постепенный переход от простейших программ к объектно-ориентированному программированию. Составление последовательности команд для решения конкретных задач на языке программирования называется разработкой программ, либо программированием. Для вызова среды программирования **PascalABC** необходимо запустить на выполнение файл **PascalABC.exe** или загрузить среду посредством ярлыка, если он существует на рабочем столе. Запустите систему программирования **PascalABC.Net**.
2. Рабочее окно **PascalABC** содержит уже знакомые нам элементы: это **Строка заголовка окна**, кнопки: **Свернуть, Развернуть, Закрывать**. Ниже находится **Строка меню**, затем **Панель инструментов**.
3. Под панелью инструментов находится **Вкладка**, т.е. та программа, которая сейчас открыта и **Рабочее окно программы**, т.е. окно в котором непосредственно будем набирать текст программы. По обе стороны от окна находятся **Полосы прокрутки**, которыми пользуются, если текст программы не вмещается в рабочее окно. Внизу экрана находится **Строка состояния**, показывающая на какой позиции стоит курсор.



4. Для того чтобы лучше ориентироваться в среде программирования **PascalABC**, разберем основные пункты меню. Первый пункт меню **Файл**. Как и других приложениях **Windows** мы видим пункт меню **Новый** (создаем новую программу), **Открыть** (открываем ранее сохраненную программу),



**Сохранить** (можем сохранить программу с расширением **pas**), **Сохранить все** (используется, если нужно сохранить несколько открытых программ), **Выход** (выйти из программы).

5. Следующий пункт меню **Правка**. Здесь находятся команды для работы с текстом программы. Можно отменить действие, восстановить действие, вырезать, копировать, вставить, найти, заменить, найти далее
6. Следующий пункт меню **Вид**. В этом пункте можно включить/выключить окна выполнения программы, окна отладки и др. Для этого нажимаем на соответствующую команду и видим, что появилось окно выполнения программы. Эти понятия для Вас являются новыми, и в процессе дальнейшего изучения **PascalABC** Вы познакомитесь с ними более подробно.
7. В пункте меню **Программа** можно начать выполнение программы. Обратите внимание на комбинации горячих клавиш. Запишите в тетрадях основные команды для выполнения и завершения программы:
  - выполнение программы: **Программа – Выполнить**, или **F9** или на Панели инструментов нажать .
  - завершение выполнения программы: **Программа – Завершить**, или **Ctrl+F2** или на Панели инструментов нажать .
  - выполнение программы по шагам. Если допущена ошибка в программе или необходимо проверить часть программы, вы выполняете её по шагам, т.е. нажимаете **F7**, и каждое нажатие этой клавиши соответствует выполнению одной конкретной команды.
8. Следующий пункт **Сервис**. В программе **PascalABC** есть встроенные задачи, чтобы просмотреть их содержимое необходимо выбрать пункт **Просмотреть задание**. Выбираем тему, задание, нажимаем просмотр и по условию мы можем составить программу, а программа **PascalABC** проверит правильность выполнения задания.
9. В пункте меню **Помощь** находится встроенный электронный учебник.
10. Поочередно войдите в указанные ниже разделы главного меню и найдите следующие команды:
  - в меню **Файл**: Новый; Открыть; Сохранить; Сохранить как...; Выход;
  - в меню **Правка**: Отменить – отменить изменение; Восстановить – вернуть изменение;
  - В меню **Программа**: Выполнить – выполнить программу; Остановить – остановить программу.

## Задание 2.

Изучите основные возможности работы в текстовом редакторе системы программирования **PascalABC**. Составьте простейшие программы.

### Методические указания по выполнению задания:

1. Наберите простейшую программу, соответствующую следующему условию задачи. Требуется ввести с клавиатуры два целых числа, найти их сумму, результат вывести на экран с поясняющим текстом.

```
Program Raschet;           //Название программы
Uses CRT;                 //Подключаемые модули
Var x, y, s:Integer;       // объявление имен переменных и их типа
Begin                     // начало исполнительской части
  Writeln('Введите два целых числа'); // написать на экране текст
  Readln(x,y);             // прочитать данные с клавиатуры и запомнить их в переменных
  s:=x+y;                  // выполнить расчет и запомнить его в переменной
  Writeln('Сумма чисел =',s); // написать на экране текст и значение переменной
End.                       // конец программы
```

2. Просмотрите текст файла, обратите внимание на структуру программы. Структура простейших программ выглядит следующим образом:

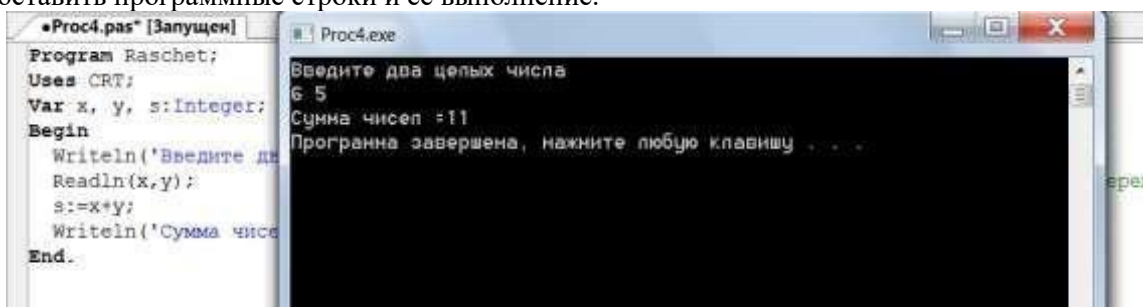
<b>Program</b>	...; заголовок программы и ее имя
<b>Var</b>	...; блок объявления переменных и их типа
<b>Begin</b>	начало исполнительской части программы
...	предложения, обеспечивающие
...	выполнение
...	программы
<b>End.</b>	конец программы (точка обязательна)

3. Программа на **PascalABC** составляется из отдельных законченных элементов, называемых предложениями. В **PascalABC** текст программы обычно начинается особым предложением –



заголовком. Заголовок необязателен. В качестве имени программы можно применять комбинацию английских букв и цифр, следует писать в одно слово и нельзя применять служебные слова языка.

4. Каждое предложение языка должно отделяться от следующего за ним точкой с запятой (;). Исключение составляют комментарии. Они не отделяются точкой с запятой.
5. Обычно каждое предложение записывается с новой строки для наглядности и более легкого понимания текста. Для этих же целей используют отступы и выравнивания.
6. Комментарии предназначены для пояснения задачи и для временного исключения из текста программы некоторых операторов. В тексте они выделяются фигурными скобками { } или отделяются двумя косыми чертами //. Комментарии игнорируются компьютером при выполнении, однако при выводе текста программы – печатаются.
7. В **PascalABC** имеется особая группа слов, таких как, например: `begin`, `for`, `end`, `program` и другие, за которыми закреплены специальные смысловые значения. Такие слова называются служебными (зарезервированными) и должны употребляться в строгом соответствии с заложенным в них смыслом.
8. Существует и другая группа имен, имеющих стандартно определенный смысл, например, `integer`, `writeln` и другие. Их так и называют – стандартные или предопределенные имена.
9. Под именем программы располагается ее декларативная часть, здесь компьютеру сообщается обо всех именах констант и переменных, определяемых программистом, и о той роли, которую эти имена должны исполнять в программе.
10. За декларативной частью следует исполнительная часть программы, обрамляемая словами-ограничителями (логическими скобками): **Begin** и **End**. Между указанной парой слов и размещаются предложения, выполняющие в программе те или иные действия. Исполнительную часть программы называют телом программы.
11. Запустите набранную программу на выполнение (поскольку в программе выполняется подключение модуля **CRT**, то запускать программу нужно сочетанием клавиш **Shift+F9**). Если после запуска программы внизу окна появляется красная строка с сообщением, то в строке, где находится курсор или в предыдущей находится ошибка. Внимательно просмотрите всю строку, найдите и исправьте ошибку. После исправления всех ошибок и появления в новом окне начала работы программы, введите нужные данные (если в программе подразумевается ввод нескольких переменных, то это следует делать через **Enter** или пробел), получите результат работы и проверьте его на правильность. Так как текст программы и ее работа показываются в разных окнах (если подключен модуль **CRT**), можно сопоставить программные строки и ее выполнение.



12. Создайте в папке своей группы папку с именем **Begin**. Сохраните набранную программу в папку **Begin** своей группы под именем **Begin1.pas**.
13. Разберитесь с работой программы и измените ее так, чтобы она вычисляла не сумму, а разность чисел. Проверьте правильность работы измененной программы. Сохраните программу под именем **Begin2.pas** в папке **Begin**.
14. Выполните команду **Файл – Новый**. Наберите текст программы. При наборе текста программы соблюдайте позиционирование (отступы) строк. Это не влияет на работу программы, но делает ее читабельной и облегчает поиск ошибок. В программе подсчитывается доход клиента за 1 год в зависимости от банковского процента и от величины денежного вклада.



```

Proc4.pas* [Запущен] •Program1.pas*
Program doxod;                               {название программы}
Var b,a:integer;                             {объявление переменных и их типа}
    c:real;
Begin                                         {начало программы}
    Writeln('Доход от вклада');              {вывод текста на экран}
    Write( 'Введите величину вклада в рублях: ' ); {вывод текста}
    Readln(b);                               {ввод целого числа в переменную b}
    Write('Введите величину банковского процента ');
    Readln(a);
    c:=a*b/100;                               {расчет значения переменной c}
    Writeln('Ваш доход =',c, ' рублей');      {вывод текста, значения переменной и текста}
end.

```

15. Запустите программу на выполнение. Введите следующие данные: введите величину вклада в рублях: 1000; введите величину банковского процента: 10. В результате должен получиться ответ: ваш доход = 100 рублей
16. Снова запустите программу и введите другие разумные исходные данные.
17. Вернитесь в текст, удалите символ «;» первой строке программы и запустите ее на исполнение. Проанализируйте сообщение об ошибке (красная строчка с сообщением).

Список ошибок

	Строка	Описание	Файл	Путь
✖	1 2	Ожидалось ';'.	Program1.pas	C:\PABCWork.NET

18. Исправьте ошибку, затем удалите точку после последнего **End** в программе. Эта ошибка часто встречается у начинающих. Запустите программу и посмотрите, как реагирует **PascalABC** на подобную ошибку.
19. Удалите любую букву, например, в слове **Writeln**. Посмотрите, как реагирует **PascalABC** на подобную ошибку.
20. Удалите в блоке **Var** объявленную переменную и посмотрите, как отреагирует **PascalABC** на запуск программы с такой ошибкой. Запоминайте сообщения компьютера. Исправьте ошибки и сохраните программу под именем **Begin3.pas** в папке **Begin**.

### Индивидуальные задания:

Составьте программу соответствующую следующей задаче:

Программа запрашивает имя пользователя и его возраст, по введенным данным определяет год рождения (текущий год запрашивается с клавиатуры), выводит его на экран и прощается с пользователем по имени.



## Практическая работа №3

### «Составление программ линейной структуры»

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с линейной структурой.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

---

#### Задание 1.

В системе программирования **PascalABC.NET** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

**Методические указания по выполнению задания:**

1. Запустите систему программирования **PascalABC.NET**.
2. В окне редактора наберите текст программы, которая вычисляет сторону и периметр квадрата, если известна его площадь.

```
Program Primer1;  
Var a,P,S:Real;  
Begin  
    Write('S='); Readln(S);  
    a:=Sqrt(S);  
    P:=4*a;  
    Writeln('a=',a:5:2,'ед. ');  
    Writeln('P=',P:5:2,'ед. ');  
End.
```

В **PascalABC** используется несколько типов представления числовых значений, на начальном этапе будут рассмотрены лишь некоторые из них: **Integer** – целые числа в интервале от -2147483648 до 2147483647; **Real** – вещественные – целые и дробные положительные и отрицательные числа

Описания констант в декларативной части производится перед переменными, и предусматривают определенную форму записи чисел (дополнительно тип константы не оговаривается): если константа записана с точкой, тип константы считается Real. При записи значения константы используется знак равенства.

Переменная – это вид объектов в программе, предназначенный для хранения информации во время выполнения программы. По правилам **PascalABC** каждая переменная должна быть объявлена, т.е. описана в декларативной части программы. Переменная не имеет какого-либо конкретного значения до тех пор, пока компьютеру не будет дано точное предписание, поместить что-либо определенное в соответствующую ячейку памяти.

Описание переменных следует за описанием констант. В описании переменных после двоеточия указывается тип переменной/

В **PascalABC** возможны следующие действия (группы операций записаны в порядке приоритета): операция возведения в степень; умножение, деление, деление целочисленное, получение остатка от целочисленного деления; сложение, вычитание.

В пределах одной группы приоритета порядок выполнения операций, если нет скобок, определяется последовательностью записи.

3. Сохраните созданную программу в папке **Begin**, созданную ранее под именем **Zad1.pas**.
4. Выполните команду файл – Новый и наберите следующую программу. Разберитесь в её работе.



```

Program Primer2;
Uses crt;
  Var  a, s, d, e : Integer;
Begin
  Writeln('Сумма цифр трехзначного числа');
  Write('Введите целое трехзначное число ');
  Readln(a);
  Clrscr;
  s:= trunc(a/100);
  d:= trunc((a-s*100)/10);
  e:=a-s*100-d*10;
  writeln('Сумма цифр трехзначного числа=', s+d+e);
End.

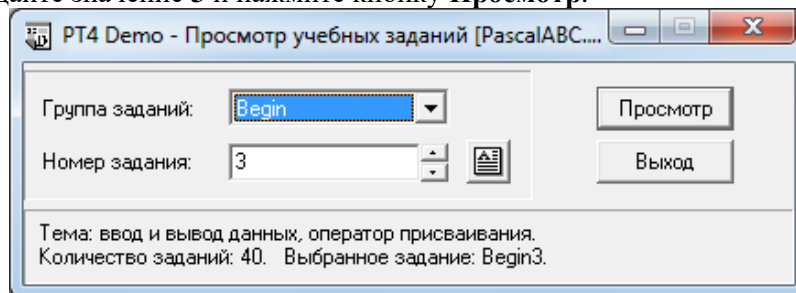
```

- Определите с помощью каких стандартных операций вычисляется количество сотен и количество десятков. Запишите в тетрадь синтаксис, используемой для этого функции.
- Измените программу так, чтобы для определения количества сотен и количества десятков использовались операции целочисленного деления и получения остатка от целочисленного деления.
- Сохраните созданную программу в папке **Begin** под именем **Zad2.pas**.

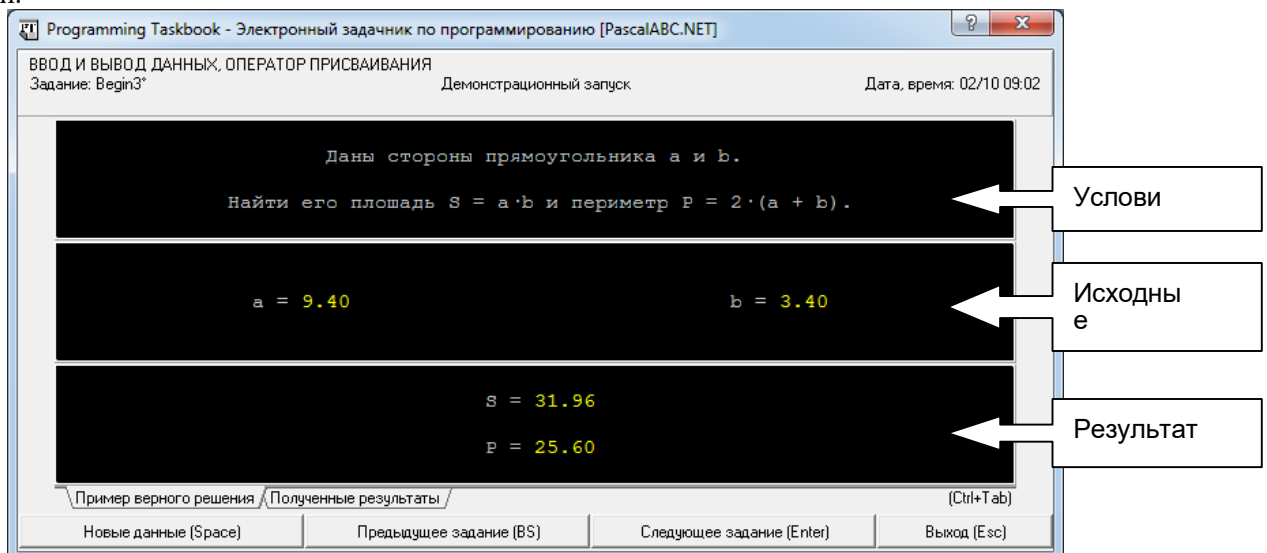
**Задание 2.** Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **Begin3**, **Begin4**, **Begin5**.

**Методические указания по выполнению задания:**

- Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле номер задания задайте значение **3** и нажмите кнопку **Просмотр**.

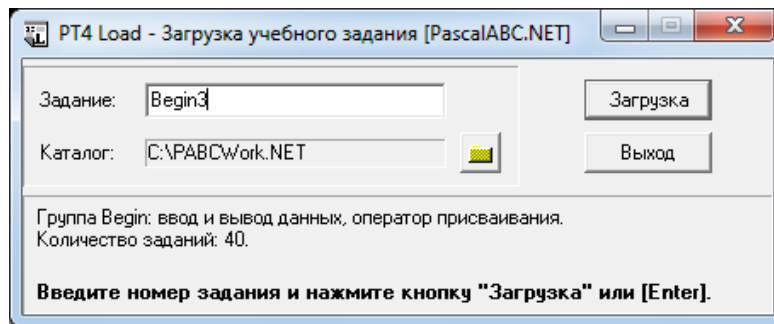


- Окно просмотра текста задачи состоит из трех разделов. В первом записано условие задачи, во втором представлены значения исходных данных, а в третьем разделе окна указаны результаты решения задачи.

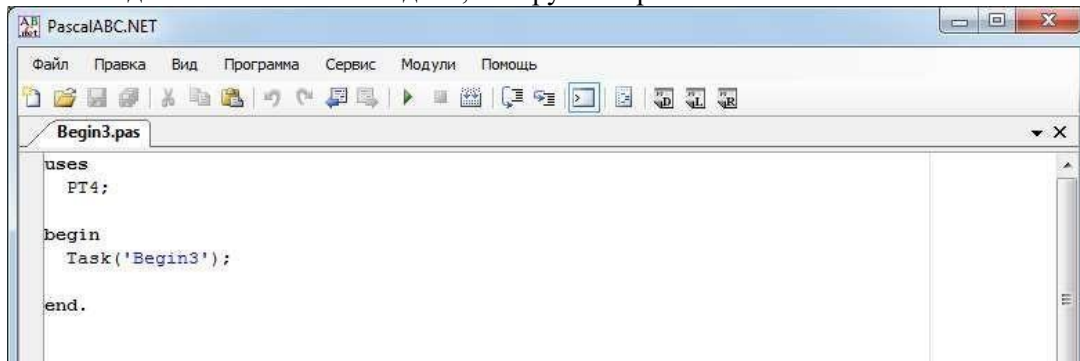


- Решим данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **begin3** и нажмите кнопку **Загрузка**.

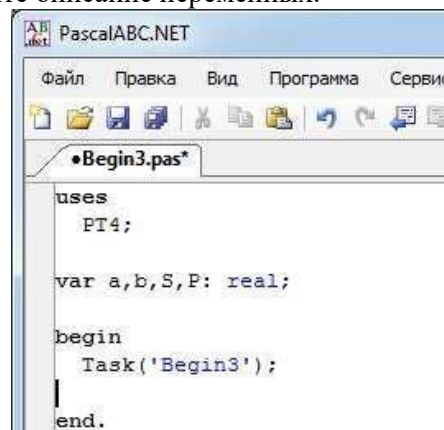




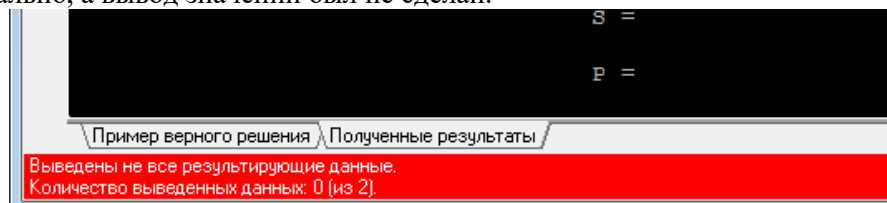
4. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.



5. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – выполнить** или нажмите соответствующую кнопку на панели инструментов.
6. При написании программы для решения задачи нам необходимо описать четыре переменные, две из них этого стороны прямоугольника **a, b**, именно их значения указаны в окне исходных данных, и переменные **S** и **P**, для результатов вычислений, их значения указаны в окне результатов. Описание переменных осуществляется в разделе **var**. Проанализировав значения исходных данных и полученных результатов в задачнике, можно сделать вывод, что переменные имеют тип **real**. Перейдите к окну шаблона программы и выполните описание переменных.



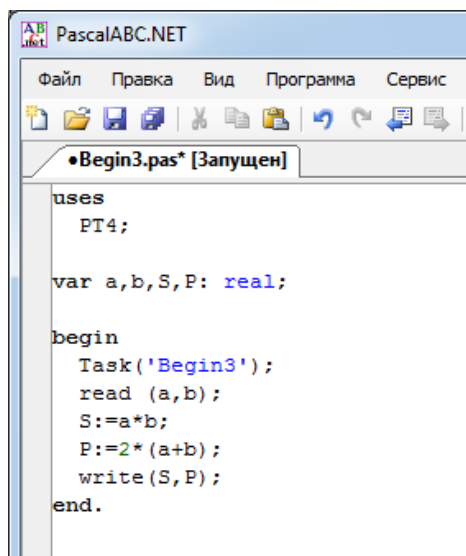
7. Далее перейдем к написанию программы. Нам необходимо ввести значения переменных **a, b**, для этого необходимо считать их значения с помощью оператора ввода **read**. Запустите программу на исполнение, программа выдаст сообщение, что **выведены не все результирующие данные**, т.е. ввод прошел нормально, а вывод значений был не сделан.



8. Для того чтобы выполнить вывод результирующих значений на экран нам необходимо выполнить расчет площади и периметра, формулы для вычисления представлены в условии задачи  $S=ab$ ,  $P=2(a+b)$ . Запишите данные формулы, используя оператор присваивания.



9. Выполните запуск программы, изменился ли текст сообщения? Почему не изменился? Какой оператор необходимо добавить в текст программы?
10. Добавьте в тело программы оператор вывода, обратите внимание на последовательность переменных при выводе.



11. Запустите программу, изменился ли текст сообщения?
12. Выполните самостоятельно решение задач **Begin4**, **Begin5**.

### Задание 3.

Составьте программы для решения следующих задач:

- Рассчитайте и выведите на экран количество рабочих часов в месяце, если продолжительность рабочего дня равна 8 часам в день, а число рабочих дней в месяце запрашивается у пользователя вашей программы.
- Скорость передачи данных по локальной сети запрашивается у пользователя и измеряется в битах в секунду. Ученик качал игру T минут (время запрашивается у пользователя). Рассчитайте и выведите на экран размер файла (в Гбайтах), который скачал ученик и сколько денег придётся заплатить ему за трафик, если первый Гбайт не оплачивается, а всё то, что сверху - по у рублей за Гбайт (запрашивается у пользователя).
- Жёсткий диск имеет объём свободного пространства X Гбайт – запрашиваемая величина. Сколько книг, каждая из которых состоит из 350 страниц, на каждой странице по 35 строк, в каждой строке по 55 символов, можно записать на жёсткий диск, если для хранения кода одного символа отводится 2 байта?

### Задание 4.

Проверьте файлы, созданные в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

## Практическая работа №4

### «Составление программ разветвляющейся структуры»

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с разветвляющейся структурой.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

### Задание 1.

В системе программирования **PascalABC.Net** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

#### Методические указания по выполнению задания:

1. Запустите систему программирования **PascalABC.Net**.
2. В папке своей группы на диске создайте папку **IF**, в дальнейшем все программы необходимо сохранять в эту папку.



3. В окне редактора наберите текст программы, которая вычисляет квадратный корень из числа. Сохраните файл под именем **Zad1.pas**.

```
•Zad1.pas
program IF1;
  var a:real;
begin
  writeln ('Введите значение числа A');
  readln(a);
  if a>=0 then
    writeln('Квадратный корень числа ',sqrt(a))
  else
    writeln('Корень из отрицательного числа не извлекается');
  writeln('Программа завершена!');
end.
```

4. Проверьте правильность работы программы, задав в качестве исходных данных **A** следующие значения: **655536; -256; 125,44**.
5. Поставьте «;» после оператора **writeln('Квадратный корень числа',sqrt(a))** в строке 7. Запустите программу на исполнение. Появилось сообщение об ошибке. Почему это произошло? Исправьте ошибку.
6. Внесите изменения в программу в строки **9-10**, добавив составной оператор **begin-end**.
- ```
begin writeln('Корень из отрицательного числа не извлекается');
writeln('Программа завершена!'); end;
```
7. Используя программу **DiagramDesigner**, начертите блок-схему, которая соответствует полученной программе. Сохраните файл с блок-схемой в папке **IF** под именем **Zad1.ddd**.
8. Исправьте программу так, чтобы в случае равенства числа нулю программа выводила на экран монитора сообщение «**Значение квадратного корня равняется нулю**». Сохраните программу.

### Задание 2.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **If6**, **If8**.

**Методические указания по выполнению задания:**

- Создайте новое окно, выполнив команду **Файл – Новый**.
- Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле группа заданий выберите **If**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
- Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **If6 (If8)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
- Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
- В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
- Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу еще два раза.
- Сохраните созданные программы в папке **IF**.

### Задание 3.

Используя систему программирования **PascalABC.NET**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **DiagramDesigner**. Созданные программу и её блок-схему сохранить в папку **IF**, в названии файла укажите номер задачи.

**Варианты заданий:**



| мер<br>варианта | 1   | 2   | 3   | 4   | 5   | 6   | 7   | 8   | 9   | 10   | 11   | 12   | 13  | 14  | 15  |
|-----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|-----|-----|-----|
| мера<br>заданий | ,25 | ,24 | ,23 | ,22 | ,21 | ,20 | ,19 | ,18 | ,17 | 0,16 | 1,15 | 2,14 | 3,1 | 4,2 | 5,3 |

#### Задания:

- Даны три действительные числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень — отрицательные.
- Даны две точки  $A(x_1, y_1)$  и  $B(x_2, y_2)$ . Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.
- Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник. Если да, то будет ли он прямоугольным.
- Даны действительные числа  $x$  и  $y$ , не равные друг другу. Меньшее из этих двух чисел заменить половиной их суммы, а большее — их удвоенным произведением.
- На плоскости  $XOY$  задана своими координатами точка  $A$ . Указать, где она расположена: на какой оси или в каком координатном угле.
- Даны целые числа  $m, n$ . Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.
- Дано двухзначное число  $N$ . Проверить, будет ли сумма его цифр четным числом.
- Дано двухзначное число  $N$ . Проверить, будет ли сумма больше 10.
- Определить, является ли треугольник со сторонами  $a, b, c$  равнобедренным.
- Определить, является ли треугольник со сторонами  $a, b, c$  равнобедренным.
- Определить, имеется ли среди чисел  $a, b, c$  хотя бы одна пара взаимно противоположных чисел.
- Подсчитать количество отрицательных чисел среди чисел  $a, b, c$ .
- Подсчитать количество положительных чисел среди чисел  $a, b, c$ .
- Подсчитать количество целых чисел среди чисел  $a, b, c$ .
- Дано целое число. Если оно является положительным, то прибавить к нему 1; в противном случае не изменять его. Вывести полученное число.
- Дано целое число. Если оно является положительным, то прибавить к нему 1; в противном случае вычесть из него 2. Вывести полученное число.
- 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.
- Даны два действительных числа. Вывести первое число, если оно больше второго, и оба числа, если это не так.
- Даны два действительных числа. Заменить первое число нулем, если оно меньше или равно второму, и оставить числа без изменения в противном случае.
- Даны три действительных числа. Выбрать из них те, которые принадлежат интервалу  $(1, 3)$ .
- Есть окружность с центром в начале координат и радиусом 5 единиц. Ввести с клавиатуры два числа, как координаты точки на той же плоскости, что и окружность. Вывести одно из сообщений: точка внутри окружности; точка на окружности; точка вне окружности.
- Найти действительные корни квадратного уравнения  $ax^2 + bx + c = 0$  для любых вводимых значений коэффициентов. Рассмотреть ситуации:  $a = 0$ ,  $D < 0$  и обычное решение с двумя корнями.
- Решить уравнение  $Ax = B$ . Предусмотреть случаи отсутствия решения и бесконечного множества решений.
- Вычислить площадь треугольника по формуле Герона для любых вводимых длин сторон  $A, B$  и  $C$ :  $p = \frac{a+b+c}{2}$ ;  $S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)}$ . Предусмотреть проверку существования треугольника.
- Даны три числа. Найти сумму двух наибольших из них.

#### Задание 4.

Проверьте файлы, которые были сохранены в папку **IF** в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.



## Практическая работа №5 «Составление программ усложненной разветвляющейся структуры»

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с усложненной разветвляющейся структурой.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

### Задание 1.

В системе программирования **PascalABC.Net** составьте программу с усложненной разветвляющейся структурой по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

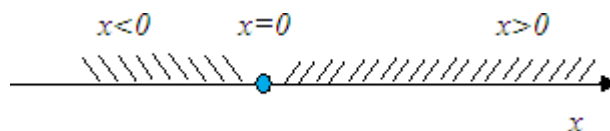
**Методические указания по выполнению задания:**

1. Запустите систему программирования **PascalABC.Net**.
2. В папке своей группы на диске создайте папку **IF2**, в дальнейшем все программы необходимо сохранять в эту папку.
3. В окне редактора наберите текст программы, которая вычисляет значение функции:

$$f(x) = \begin{cases} x - 2, & x > 0 \\ 5, & x = 0 \\ x^2, & x < 0 \end{cases}$$

При решении задач часто приходится рассматривать не два, а большее количество вариантов. Это можно реализовать, используя несколько условных операторов. В этом случае после служебных слов **Then** и **Else** записывается новый условный оператор.

Для решения этой задачи рассмотрим координатную прямую, на которой отметим промежутки, на которые разбиваются все значения переменной  $x$ .



Начнем записывать условный оператор:

ли

то

вычислить  $y$  по формуле  
 $y = x - 12$

Что же должно выполняться в случае иначе? На эту ветку оператора попадают все не положительные значения  $x$ . Если бы для этих чисел нужно было бы выполнить один и тот же оператор (или группу операторов), то проблемы бы не стояло. Но нам нужно этот промежуток разделить еще на две части (отрицательные и ноль), и для части выполнить свой оператор. Поэтому ветка **Иначе** будет содержать еще один условный оператор и наш вложенный условный оператор будет иметь вид:



ли

```
то  
вычислить y по формуле  $y=x-12$  иначе  
  
если  
x=0 то  
y вычислить по формуле  $y=5$   
иначе  
y вычислить по формуле  
 $y=\text{sqr}(x)$ .
```

Тогда фрагмент программы для решения этой задачи на языке программирования **PascalABC** будет выглядеть так:

```
If x>0  
Then  
    y := x-12  
Else  
    If x=0  
    Then  
        y := 5  
    Else  
        y := sqr(x);
```

4. Дополните фрагмент описанием необходимых переменных, операторами ввода-вывода. Сохраните программу в папку **IF2** под именем **Zad1.pas**. Протестируйте работу программы при различных значениях аргумента, так, чтобы можно было проверить каждую из ветвей.

Когда оператор **If** появляется внутри другого оператора **If**, они считаются вложенными. Такое вложение используется для уменьшения числа необходимых проверок. Этот метод часто обеспечивает большую эффективность, однако одновременно он уменьшает наглядность программы. Не рекомендуется использовать более одного-двух уровней вложения **If**. За вторым уровнем вложения становится трудно восстановить последовательность проверки условий каждым условным оператором.

5. Создайте новую вкладку. В окне редактора наберите текст программы, которая выполняет случайное предсказание одного из десяти вариантов ближайшего будущего с вероятностью 1/20, в остальных случаях программа выводит—«неудача».

Для написания данной программы воспользуемся оператором выбора. Данный оператор служит для выбора одного из помеченных вариантов действия (операторов), в зависимости от значения «параметра».

В программе будем использовать функцию **Random(x)**, которая генерирует случайное число, с равномерной плотностью распределения на заданном интервале. Для инициализации распределения в начале программы необходимо вызвать процедуру **Randomize**.



```

Program2.pas*
Program FUTURE;
Var N : Word;
Begin
  Writeln('ПРЕДСКАЗАНИЕ БУДУЩЕГО');
  Randomize;
  N:=Random(20)+1;      { N - случайное число от 1 до 20 }
  Writeln;  write('Вас ожидает ');
  Case N of
    1 : writeln('счастье');
    2 : writeln('пятерка');
    3 : writeln('дорога');
    4 : writeln('двойка');
    5 : writeln('болезнь');
    6 : writeln('здоровье');
    7 : writeln('деньги');
    8 : writeln('любовь');
    9 : writeln('встреча');
    10 : writeln('дети')
      Else writeln('неудача')
  End;
  Writeln('Нажми Enter');
  Readln;
End.

```

6. Сохраните программу в папку **IF2** под именем **Zad2.pas**.

### Задание 2.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **If26**, **Case10**.

**Методические указания по выполнению задания:**

1. Создайте новое окно, выполнив команду **Файл – Новый**.
2. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле группа заданий выберите **If**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
3. Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **If26 (Case10)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
4. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
5. В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
6. Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу еще несколько раз.
7. Сохраните созданные программы в папке **IF2**.

### Задание 3.

Используя систему программирования **PascalABC.NET**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **Diagram Designer**. Созданные программу и её блок-схему сохранить в папку **IF**, в названии файла укажите номер задачи.

**Варианты заданий:**

| Номер<br>варианта | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|-------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|
|-------------------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|

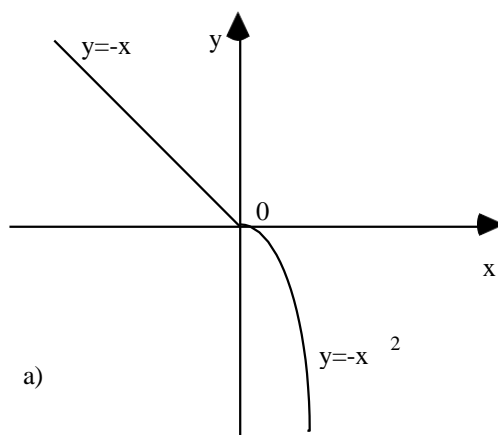


|                |     |     |     |     |     |     |     |     |     |      |      |      |      |      |      |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|
| Номера заданий | ,16 | ,17 | ,18 | ,19 | ,20 | ,21 | ,22 | ,23 | ,24 | 0,25 | 1,26 | 2,27 | 3,28 | 4,29 | 5,30 |
|----------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|------|

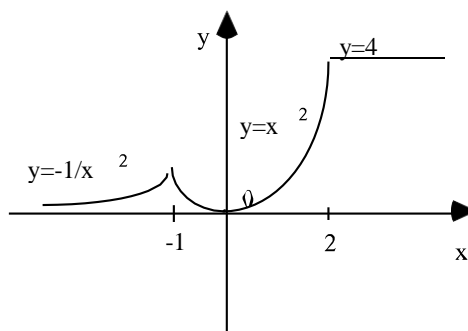
### Задания:

1. Напишите программу, которая определяет, попадает ли точка  $A$  с координатами  $(x, y)$  внутрь круга радиуса  $R$ . Центр круга совпадает с началом координат.
2. Напишите программу «Угадай число». Компьютер «загадывает» число, а пользователь его отгадывает.
3. Напишите программу, которая находит сумму положительных чисел, больших 20, меньших 100 и кратных 3.
4. Составьте программу для упорядочения трёх чисел  $a, b, c$  по возрастанию таким образом, чтобы имени  $a$  соответствовало наименьшее число, имени  $b$  - среднее, имени  $c$  - наибольшее.
5. Составьте программу для упорядочения трех чисел  $a, b, c$  по возрастанию таким образом, чтобы имени  $a$  соответствовало наименьшее число, имени  $b$  - среднее, имени  $c$  - наибольшее.
6. Напишите программу, которая преобразовывает римские числа в натуральные числа.
7. Дано действительное число  $a$ . Вычислить  $f(a)$ , если
 
$$f(a) = \begin{cases} x^2, & -2 \leq x \leq 2 \\ 4, & \text{в противном случае} \end{cases}$$
8. Дано действительное число  $a$ . Вычислить  $f(a)$ , если
 
$$f(a) = \begin{cases} x^2 + 4x + 5, & x \leq 2 \\ \frac{1}{x^2 + 4x + 5}, & \text{в противном случае} \end{cases}$$
9. Дано действительное число  $a$ . Вычислить  $f(a)$ , если
 
$$f(a) = \begin{cases} 0, & x \leq 0 \\ x, & 0 < x \leq 1 \\ x^4, & \text{в остальных случаях} \end{cases}$$
10. Дано действительное число  $a$ . Вычислить  $f(a)$ , если
 
$$f(a) = \begin{cases} 0, & x \leq 0 \\ x^2 - x, & 0 < x \leq 1 \\ x^2 - \sin x^2, & \text{в остальных случаях} \end{cases}$$
11. Даны действительные положительные числа  $a, b, c, d$ . Выяснить, можно ли прямоугольник со сторонами  $a$  и  $b$  уместить внутри прямоугольника со сторонами  $c$  и  $d$  так, чтобы каждая из сторон одного прямоугольника была параллельна каждой стороне второго прямоугольника;
12. Даны действительные положительные числа  $a, b, c, x, y$ . Выяснить, пройдет ли кирпич с ребрами  $a, b, c$  в прямоугольное отверстие со сторонами  $x, y$ . Просовывать кирпич в отверстие разрешается только так, чтобы каждое из его ребер было параллельно или перпендикулярно каждой из сторон отверстия.
13. Дано действительное число  $a$ . Для функций  $f(x)$ , графики которых представлены на рисунках, вычислить  $f(a)$





14. Дано действительное число  $a$ . Для функций  $f(x)$ , графики которых представлены на рисунках, вычислить  $f(a)$



15. Даны целые числа  $a, b, c$ . Если  $a \leq b \leq c$ , то все числа заменить их квадратами, если  $a > b > c$ , то каждое число заменить наибольшим из них, в противном случае сменить знак каждого числа.
16. Составить программу, позволяющую получить словесное описание школьных отметок (1 — «плохо», 2 — «неудовлетворительно», 3 — «удовлетворительно», 4 — «хорошо», 5 — «отлично»).
17. Написать программу, классифицирующую треугольники (остроугольные, прямоугольные, тупоугольные), если даны углы.
18. Написать программу, которая по номеру дня недели - целому числу от 1 до 7 выдает в качестве результата количество пар в соответствующий день.
19. Написать программу нахождения числа дней в месяце, если даны: Номер месяца  $n$  - целое число  $a$ , равное 1 для високосного года и равное 0 в противном случае.
20. Написать программу, которая по номеру дня недели выводит его название.
21. В зависимости от того введена ли открытая скобка или закрытая, напечатать «открытая круглая скобка» или «закрытая фигурная скобка».
22. В зависимости от введенного символа  $L, S, V$  программа должна вычислять длину окружности; площадь круга; объем цилиндра.
23. Написать программу, которая по введенному числу от 0 до 15 выводит название цвета, соответствующего этому коду.
24. Определить, является ли введенная буква русского алфавита гласной.
25. Напишите программу, которая по введенному числу из промежутка  $0..24$ , определяет время суток.
26. Напишите программу, которая по введенному номеру месяца високосного или невисокосного года, выводит количество дней в месяце.
27. Написать программу, которая по номеру месяца выдает название следующего за ним месяца (при  $t = 1$  получаем февраль, 4 — май и т.д.).
28. Для каждой введенной цифры (0 — 9) вывести соответствующее ей название на английском языке (0 — zero, 1 — one, 2 — two, ...).
29. Написать программу, позволяющую по последней цифре числа определить последнюю цифру его квадрата.
30. Написать программу, которая сообщает сдал студент экзамен или нет. Если оценка 3, 4, 5 - то экзамен сдан; если оценка 2, то не сдан.

#### Задание 4.



Проверьте файлы, которые были сохранены в папку **IF** в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

## Практическая работа №6

### «Составление программ циклической структуры»

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с циклической структурой.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

---

#### Задание 1.

В системе программирования **PascalABC.Net** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

#### Методические указания по выполнению задания:

1. Запустите систему программирования **PascalABC.Net**.
2. В папке своей группы на диске создайте папку **Циклы**, в дальнейшем все программы необходимо сохранять в эту папку.
3. В окне редактора наберите текст программы, которая из чисел от 10 до 99 выводит на экран те, сумма цифр которых равна  $n$  ( $0 < n \leq 18$ ). Сохраните файл под именем **Zad1.pas**.

```
Program PR_1;
Var s, k, n, p1, p2: Integer;
Begin
    Writeln ('Введите целое число ');
    Readln (n);
    For k:= 10 to 99 do
        Begin
            P1:= k div 10;
            P2:= k mod 10;
            S:= p1+p2;
            If s=n Then writeln (k);
        End;
    End.
```

4. Проанализируйте текст программы и ответьте на вопросы:
  - Каким образом выделили последнюю (младшую) цифру числа?
  - Каким действием в программе выделили первую (старшую) цифру числа?
5. Объясните обозначение переменных.
6. Рассмотрим выполнение программы в пошаговом режиме для  $n = 2$ . Режим пошагового выполнения предназначен для отладки программы. Для выполнения одного шага (одной строки) программы следует нажать клавишу **F8** (шаг без входа в подпрограмму), либо клавишу **F7** (шаг со входом в подпрограмму). Окно отладки позволяет просматривать во время пошагового исполнения программы значения переменных. По умолчанию оно располагается в правом верхнем углу окна редактора. Для добавления переменной или выражения в окно отладки следует нажать комбинацию клавиш **Ctrl-F5** или выполнить команду **Программа – Добавить переменную**. Добавьте в окно отладки переменные **k, p1, p2** и **s** и запустите программу в пошаговом режиме.



| Окно отладки |          |         |
|--------------|----------|---------|
| Выражение    | Значение | Тип     |
| k            | 32       | integer |
| p1           | 3        | integer |
| p2           | 1        | integer |
| s            | 4        | integer |
|              |          |         |

7. В результате работы программы на экране появится 2 числа: 11, 20.
8. В данной программе используется цикл со счетчиком, изменим программу так чтобы в программе использовался цикл с предусловием. Для этого необходимо внести изменения в строки программы **6-11**.

```

Readln (n);
k:=10;
while k<>99 do
begin
    P1:= k div 10;
    P2:= k mod 10;
    S:= p1+p2;
    If s=n Then writeln (k);
    k:=k+1;
end;

```

9. Запустите программу на исполнение при  $n = 2$ . Проанализируйте полученные результаты.
10. Самостоятельно выполните изменение программы так, чтобы в ней использовался цикл с постусловием.

### Задание 2.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **While1, While4, While11, For5, For12**.

#### Методические указания по выполнению задания:

1. Создайте новое окно, выполнив команду **Файл – Новый**.
2. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле группа заданий выберите **While (For)**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
3. Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **While1 (While4, While11, For5, For12)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
4. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
5. В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
6. Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу необходимое количество раз.
7. Сохраните созданные программы в папке **Циклы**.

### Задание 3.

Используя систему программирования **PascalABC.NET**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **Diagram Designer**. Созданные программу и её блок-схему сохранить в папку **Циклы**, в названии файла указать номер задачи. Каждый студент должен выполнить все задания.

- С помощью оператора **while** напишите программу определения суммы всех нечетных чисел в диапазоне от 1 до 99 включительно.
- Найти нечетные и кратные трем числа в диапазоне от 30 до 60 включительно. Распечатать их в порядке убывания.



- В сберкассе на трехпроцентный вклад положили S рублей. Какой станет сумма вклада через N лет?

#### Задание 4.

Проверьте файлы, которые были сохранены в папку **Циклы** в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

### Практическая работа №7

#### Составление программ усложненной циклической структуры

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с усложненной циклической структурой.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

#### Задание 1.

В среде программирования **Visual Studio** составьте программу с вложенными циклами. Протестируйте работу программы. Выполните задания на модификацию созданных программ.

#### Методические указания по выполнению задания:

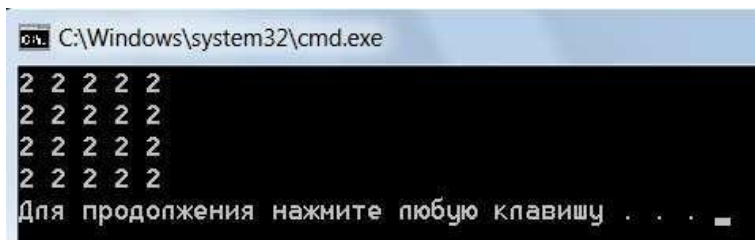
1. Запустите среду программирования **Visual Studio**.
2. В папке своей группы на диске создайте папку **Циклы2**, в дальнейшем все программы необходимо сохранять в эту папку.
3. Создайте новое консольное приложение с именем **Zad1**. В окне редактора наберите текст программы, которая позволит вывести на экран числа следующим образом:

```
2  2  2  2  2
2  2  2  2  2
2  2  2  2  2
2  2  2  2  2
```

Циклы могут быть простые или вложенные (кратные, циклы в цикле). Вложенными могут быть циклы любых типов: **while**, **do while**, **for**. Каждый внутренний цикл должен быть полностью вложен во все внешние циклы. «Пересечения» циклов не допускаются.

```
static void Main()
{
    for (int i = 1; i <= 4; i++)
    {
        for (int j = 1; j <= 5; j++)
        {
            Console.Write("2 ");
        }
        Console.WriteLine();
    }
}
```

4. Проверьте правильность работы программы, задав в качестве исходных данных следующие значения:





- Измените программу так, чтобы таблица содержала  $n$  строк и  $m$  столбцов (значения  $n$  и  $m$  вводятся с клавиатуры).
- Создайте новое консольное приложение **Zad2**. В окне редактора текста напишите программу, которая выводит на экран числа в виде следующей таблицы:

```

C:\Windows\system32\cmd.exe
n= 7
1
3
2 2
4 4
3 3 3
5 5 5
4 4 4 4
6 6 6 6
5 5 5 5 5
7 7 7 7 7
6 6 6 6 6 6
8 8 8 8 8 8
7 7 7 7 7 7 7
9 9 9 9 9 9 9

```

- Протестируйте работу программы. Сохраните программу.

## Задание 2.

В среде программирования **Visual Studio** составьте программу с оператором цикла и условным оператором.

Протестируйте работу программ. Выполните задания на модификацию созданных программ.

### Методические указания по выполнению задания:

- Создайте новое консольное приложение с именем **Zad3**. В окне редактора наберите текст программы, которая выводит на экран таблицу значений функции  $y(x)$  для  $x \in [a, b]$  с шагом  $h$ .

$$y(x) = \begin{cases} (x^3 + 1)^2, & \text{if } x < 0 \\ 0, & \text{if } 0 \leq x < 1 \\ x^2 - 5x + 1, & \text{if } x \geq 1 \end{cases}$$



```

static void Main(string[] args)
{
    Console.Write("a= ");
    double a = double.Parse(Console.ReadLine());
    Console.Write("b= ");
    double b = double.Parse(Console.ReadLine());
    Console.Write("h= ");
    double h = double.Parse(Console.ReadLine());
    double y;
    int i = 1;
    Console.WriteLine("{0,3} {1,5} {1,5}", "#", "x", "f(x)");
    for (double x = a; x <= b; x += h, i++)
    {
        if (x < 0)
        {
            y = Math.Pow(Math.Pow(x, 3) + 1, 2);
        }
        else
        {
            if (x < 1)
            {
                y = 0;
            }
            else
            {
                y = Math.Abs(x * x - 5 * x + 1);
            }
        }
        Console.WriteLine("{0,3} {1,5:f2} {2,5:f2}", i, x, y);
    }
}

```

2. Проверьте правильность работы программы.

```

C:\Windows\system32\cmd.exe
a= 1
b= 5
h= 2
#    x    x
1  1,00  3,00
2  3,00  5,00
3  5,00  1,00

```

### Задание 3.

В среде программирования **Visual Studio** составьте программу реализующую алгоритм расчета простейшей рекуррентной последовательности. Протестируйте работу программы. Выполните задания на модификацию созданных программ.

#### Методические указания по выполнению задания:

1. Создайте новое консольное приложение с именем **Zad4**. В окне редактора наберите текст программы,

в которой вычисляются первые  $n$  членов арифметической прогрессии при условии, что  $a_1 = \frac{1}{2}$  и

$$d = \frac{1}{4}.$$

Пусть  $a_1, a_2, \dots, a_n$  — произвольная числовая последовательность. Рекуррентным соотношением называется такое соотношение между членами последовательности, в котором каждый следующий член выражается через несколько предыдущих, т.е.:



$$a_k = f(a_{k-1}, a_{k-2}, \dots, a_{k-l}), k > l$$

Самым простым примером рекуррентной последовательности является арифметическая прогрессия.

Рекуррентное соотношение для нее записывается в виде  $a_k = a_{k-1} + d$ , где  $d$  – разность прогрессии.

Зная первый элемент и разность прогрессии, и, используя данное рекуррентное соотношение, можно последовательно вычислить все остальные члены прогрессии.

```
static void Main(string[] args)
{
    double a = 0.5;
    const double d = 0.25;
    Console.WriteLine("n=");
    int n = int.Parse(Console.ReadLine());

    //вывели первый член последовательности
    Console.WriteLine("a1={0}", a);

    //организуем вычисление 2, 3, ... ,n члена последовательности
    for (int i = 2; i <= n; i++)
    {
        //для этого прибавляем к предыдущему члену значение d
        a += d;
        //и выводим новое значение a на экран
        Console.WriteLine("a{0}={1}", i, a);
    }
}
```

2. Проверьте правильность работы программы.

```
cmd.exe C:\Windows\system32\cmd.exe
n=5
a1=0,5
a2=0,75
a3=1
a4=1,25
a5=1,5
```

#### Задание 4.

Используя среду программирования **Visual Studio**, выполнить индивидуальные задания. Созданную программу сохранить в папку **Циклы2**, в названии файла укажите номер задачи.

#### Варианты заданий:

| Номер компьютера | 1   | 2   | 3   | 4   | 5   | 6   | 7  | 8  | 9  | 10  | 11  | 12  | 13  | 14  | 15  |
|------------------|-----|-----|-----|-----|-----|-----|----|----|----|-----|-----|-----|-----|-----|-----|
| Номера заданий   | ,15 | ,14 | ,13 | ,12 | ,11 | ,10 | ,9 | ,8 | ,7 | 0,6 | 1,5 | 2,4 | 3,3 | 4,2 | 5,1 |

#### Задание 1:

Вывести на экран числа следующим образом:



|     |     |    |    |     |    |   |     |    |   |   |     |    |     |     |    |    |     |    |
|-----|-----|----|----|-----|----|---|-----|----|---|---|-----|----|-----|-----|----|----|-----|----|
| 1.  | 1   | 1  | 1  | 1   | 1  | 1 | 2.  | 1  | 2 | 3 | ... | 10 | 3.  | -10 | -9 | -8 | ... | 12 |
|     | 2   | 2  | 2  | 2   | 2  | 2 |     | 1  | 2 | 3 | ... | 10 |     | -10 | -9 | -8 | ... | 12 |
|     | 3   | 3  | 3  | 3   | 3  | 3 |     | 1  | 2 | 3 | ... | 10 |     | -10 | -9 | -8 | ... | 12 |
|     | 4   | 4  | 4  | 4   | 4  | 4 |     | 1  | 2 | 3 | ... | 10 |     | -10 | -9 | -8 | ... | 12 |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     | -10 | -9 | -8 | ... | 12 |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     |     |    |    |     |    |
| 4.  | 41  | 42 | 43 | ... | 50 |   | 5.  | 5  |   |   |     |    | 6.  | 1   | 1  | 1  | 1   | 1  |
|     | 51  | 52 | 53 | ... | 60 |   |     | 5  | 5 |   |     |    |     | 1   | 1  | 1  | 1   | 1  |
|     | 61  | 62 | 63 | ... | 70 |   |     | 5  | 5 | 5 |     |    |     | 1   | 1  | 1  | 1   | 1  |
|     | ... |    |    |     |    |   |     | 5  | 5 | 5 | 5   |    |     | 1   | 1  | 1  | 1   | 1  |
|     | 71  | 72 | 73 | ... | 80 |   |     | 5  | 5 | 5 | 5   | 5  |     | 1   | 1  | 1  | 1   | 1  |
| 7.  | 1   |    |    |     |    |   | 8.  | 6  | 6 | 6 | 6   | 6  | 9.  | 7   |    |    |     |    |
|     | 2   | 2  |    |     |    |   |     | 7  | 7 | 7 | 7   | 7  |     | 6   | 6  |    |     |    |
|     | 3   | 3  | 3  |     |    |   |     | 8  | 8 | 8 |     |    |     | 5   | 5  | 5  |     |    |
|     | 4   | 4  | 4  | 4   |    |   |     | 9  | 9 |   |     |    |     | 4   | 4  | 4  | 4   |    |
|     | 5   | 5  | 5  | 5   | 5  |   |     | 10 |   |   |     |    |     | 3   | 3  | 3  | 3   | 3  |
| 10. | 8   | 8  | 8  | 8   | 8  |   | 11. | 1  |   |   |     |    | 12. | 1   |    |    |     |    |
|     | 7   | 7  | 7  | 7   | 7  |   |     | 1  | 2 |   |     |    |     | 2   | 1  |    |     |    |
|     | 6   | 6  | 6  | 6   | 6  |   |     | 1  | 2 | 3 |     |    |     | 3   | 2  | 1  |     |    |
|     | 5   | 5  |    |     |    |   |     | 1  | 2 | 3 | 4   |    |     | 4   | 3  | 2  | 1   |    |
|     | 4   |    |    |     |    |   |     | 1  | 2 | 3 | 4   | 5  |     | 5   | 4  | 3  | 2   | 1  |
| 13. | 0   | 1  | 2  | 3   | 4  |   | 14. | 4  | 3 | 2 | 1   | 0  | 15. | 1   |    |    |     |    |
|     | 0   | 1  | 2  | 3   |    |   |     | 3  | 2 | 1 | 0   |    |     | 0   |    |    |     |    |
|     | 0   | 1  | 2  |     |    |   |     | 2  | 1 | 0 |     |    |     | 2   | 2  |    |     |    |
|     | 0   | 1  |    |     |    |   |     | 1  | 0 |   |     |    |     | 0   | 0  |    |     |    |
|     | 0   |    |    |     |    |   |     | 0  |   |   |     |    |     | 3   | 3  | 3  |     |    |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     | 0   | 0  | 0  |     |    |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     | 4   | 4  | 4  | 4   |    |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     | 0   | 0  | 0  | 0   |    |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     | 5   | 5  | 5  | 5   | 5  |
|     |     |    |    |     |    |   |     |    |   |   |     |    |     | 0   | 0  | 0  | 0   | 0  |

## Задание 2:

Постройте таблицу значений функции  $y(x)$  для  $x \in [a, b]$  с шагом  $h$ :

$$1. y(x) = \begin{cases} \overline{(0,1+x)^2} & , \text{при } x \geq 0,9 \\ 0,2x + 0,1, & \text{при } 0 \leq x < 0,9 \\ x^2 + 0,2, & \text{при } x < 0 \end{cases}$$

$$3. y(x) = \begin{cases} -4, & \text{при } x < 0 \\ x^2 + 3x + 4, & \text{при } 0 \leq x < 1 \\ 2, & \text{при } x \geq 1 \end{cases}$$

$$5. y(x) = \begin{cases} x^2 + 5, & \text{при } x \leq 5 \\ 0, & \text{при } 5 < x < 20 \\ 1, & \text{при } x \geq 20 \end{cases}$$

$$7. y(x) = \begin{cases} x + 5, & \text{при } x > 5 \\ 1, & \text{при } x = 5 \end{cases}$$

$$2. y(x) = \begin{cases} 0, & \text{при } x < a \\ x - a, & \text{при } x > a \\ x + a, & \text{при } x = a \end{cases}$$

$$4. y(x) = \begin{cases} (x^2 - 1)^2, & \text{при } x < 1 \\ 1, & \text{при } x > 1 \\ (1+x)^2, & \text{при } x = 1 \end{cases}$$

$$6. y(x) = \begin{cases} x, & \text{при } x > 0 \\ 0, & \text{при } -1 \leq x \leq 0 \\ x^2, & \text{при } x < -1 \end{cases}$$

$$8. y(x) = \begin{cases} \overline{\square\square}^1, & \text{при } x > b \\ 0, & \text{при } x = b \end{cases}$$



$$9. \quad y(x) = \begin{cases} 0, & \text{при } x < 0 \\ x^2 + 1, & \text{при } 0 \leq x < 1 \\ 1, & \text{при } x = 1 \end{cases}$$

$$11. \quad y(x) = \begin{cases} x^2 - 1, & \text{при } x \leq 1 \\ 2x - 1, & \text{при } 1 < x \leq 2 \\ x^5 - 1, & \text{при } x > 2 \end{cases}$$

$$13. \quad y(x) = \begin{cases} x^2, & \text{при } x < 1 \\ 1, & \text{при } x > 1 \\ \frac{1}{x+2}, & \end{cases}$$

$$15. \quad y(x) = \begin{cases} x+2, & \text{при } x = 1 \\ x^3 - 0,1, & \text{при } x \leq 5 \\ 0,2x - 0,1, & \text{при } 5 < x < 20 \\ x^3 + 0,1, & \text{при } x \geq 20 \end{cases}$$

$$10. \quad y(x) = \begin{cases} x^2 - 0,3, & \text{при } x < 3 \\ 0, & \text{при } 3 \leq x \leq 5 \end{cases}$$

$$12. \quad y(x) = \begin{cases} x^2 + 1, & \text{при } x > 5 \\ x^2, & \text{при } x < 1 \\ 1, & \text{при } x > 1 \\ \frac{1}{x^2 - 1}, & \end{cases}$$

$$14. \quad y(x) = \begin{cases} 0, & \text{при } x = 1 \\ x^3 - 1, & \text{при } x < 0,1 \\ 2x - 1, & \text{при } x > 0,1 \\ 0, & \text{при } x = 0,1 \end{cases}$$

### Практическая работа №8 «Обработка одномерных массивов»

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с использованием одномерных массивов.

**Оборудование, технические и программные средства:** персональный компьютер, система программирования **PascalABC.NET**.

#### Задание 1.

В системе программирования **PascalABC.Net** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

**Методические указания по выполнению задания:**

1. Запустите систему программирования **PascalABC.Net**.
2. В папке своей группы на диске создайте папку **Array1**, в дальнейшем все программы необходимо сохранять в эту папку.
3. На языке программирования **PascalABC.Net** составьте программу, которая находит сумму 30 целых чисел. Сохраните файл под именем **Zad1.pas**.

**Одномерный массив** – это фиксированное количество элементов одного и того же типа, объединенных одним именем, где каждый элемент имеет свой номер. Обращение к элементам массива осуществляется с помощью указания имени массива и номеров элементов. Нам для работы требуется массив из 30 целых чисел.

Опишем в разделе описания типов свой тип – одномерный массив, состоящий из 30 целых чисел.

```
Type MyArray = Array [1..30] Of Integer;
```

Напомним, что раздел типов начинается со служебного слова **Type**, после этого идет имя нового типа и его описание. Между именем типа и его описанием ставится знак «равно». В нашем случае **MyArray** – имя нового типа данных; **Array** – служебное слово (в переводе с английского означает «массив», «набор»); **[1..30]** – в квадратных скобках указывается номер первого элемента, затем, после двух точек, номер последнего элемента массива; **Of** – служебное слово (в переводе с английского «из»); **Integer** – тип элементов массива.

Далее необходимо заполнить наш массив числами, для этого необходимо воспользоваться циклической конструкцией следующего вида:

```
WriteLn ('Введите ', n, ' чисел');
For i:=1 To n Do ReadLn(A[i]);
```

В данном случае числа вводятся пользователем с клавиатуры.



Далее необходимо воспользоваться циклом для определения суммы чисел:

```
s:=0;  
For i:=1 To n Do s:=s+A[i];  
WriteLn ('Их сумма равна ' ,s) ;
```

Тогда решение данной задачи с использованием массива имеет вид:

```
•Zad1.pas  
Program Zad1;  
Const n=30 ;  
Type MyArray=Array[1..n] Of Integer;  
Var  
  A: MyArray;  
  s,i: Integer;  
Begin  
  WriteLn ('Введите ' ,n, ' чисел');  
  For i:=1 To n Do ReadLn(A[i]);  
  s:=0;  
  For i:=1 To n Do s:=s+A[i];  
  WriteLn ('Их сумма равна ' ,s) ;  
End.
```

4. Проверьте правильность работы программы.
5. Измените программу так, чтобы определялась сумма элементов массива, кратных заданному числу. Сохраните полученную программу в папке **Array1** под именем **Zad2.pas**.

Решение задачи изменится незначительным образом. Добавляется описание еще одной переменной для хранения значения числа, на кратность которому проверяются значения элементов массива.

Появляются операторы:

```
WriteLn ('Введите число'); ReadLn (k);
```

и изменяется оператор из тела цикла:

```
For i:=1 To n Do  
  If A[i] Mod k = 0 Then s:=s+A[i];
```

6. Измените программу так, чтобы определялось количество положительных и отрицательных элементов в данном массиве. Сохраните полученную программу в папке **Array1** под именем **Zad3.pas**.

Суть основного изменения программы заключается во введении двух переменных (счетчиков – **pos**, **neg**) для хранения значений количества положительных и отрицательных элементов в массиве.

```
Var  
  A: MyArray;  
  s,i,k: Integer;  
  pos, neg: Integer;
```

Добавьте в программу следующие строки программного кода:

```
s:=0;  
pos:=0;  
neg:=0;  
For i:=1 To n Do  
  If A[i]>0 Then Inc(pos) Else  
    If A[i]<0 Then Inc (neg);  
WriteLn (pos:4, neg:4);
```

7. Измените программу так, чтобы в массив **B** записывались номера четных элементов массива **A**. Сохраните полученную программу в папке **Array1** под именем **Zad4.pas**.

Введение массива **B** и работа с ним требует введения переменной **j** для обращения к элементам **B**. Суть решения заключается в просмотре элементов массива **A**, выявлении четных элементов и записи их номеров по текущему значению переменной **j** в массив **B**.

Решение данной задачи имеет следующий вид:



```

Zad1.pas*   •Program1.pas*
Program Zad4;
Const n=10;
Type MyArray=Array[1..n] Of Integer;
Var A, B: MyArray;
    i, j: Integer;
Begin
  WriteLn ('Введите ', n, ' чисел');
  For i:=1 To n Do ReadLn(A[i]);
  j:=0;
  For i:=1 To n Do
    If A[i] Mod 2 = 0 Then
      Begin
        Inc(j);
        B[j]:=i;
      End;
  For i:=1 To j Do
    Write(B[i]:3);
  WriteLn;
End.

```

8. Выполните команду **Файл – Новый** и составьте программу нахождения значения и номера всех отрицательных элементов. Сохраните полученную программу в папке **Array1** под именем **Zad5.pas**.

```

Zad1.pas*   Program1.pas*   •Program2.pas*
Program Zad5;
Const n=7;
Type MyArray=Array[1..n] Of Integer;
Var
  A:MyArray; i:Integer;
Begin
  WriteLn('Ввод элементов массива. Не забудьте об отрицательных элементах. ');
  For i:=1 To n Do Read(A[i]) ;
  WriteLn('Вывод отрицательных элементов массива и их номеров (индексов) ');
  For i:=1 To n do
    If A[i]<0 Then WriteLn (A[i], ' ', i, ' ');
  End.

```

9. Текст программы поиска номеров всех отрицательных элементов в массиве содержит строки: **For i:=1 To n do** Измените во второй строке **n** на **n+1** и поставьте перед текстом программы директиву компилятора **{\$R+}**. Директивы компилятора управляют режимом компиляции. Директива начинается с символа **\$** после открывающей скобки комментария, за которым следует имя директивы (состоящее из одной или нескольких букв), определяющее ее назначение. Контроль вхождения в диапазон осуществляется с помощью директивы **{\$R+}** (**{\$R-}**), при этом устанавливается или отменяется генерация кода контроля вхождения в диапазон. В режиме **{\$R+}** все индексы массивов и строк контролируются на выход за границы, а все присваивания значений скалярных переменных и ограниченных типов проверяются на вхождение в диапазон. При выходе за границы выполнение программы прекращается и выдается сообщение об ошибке.
10. Запустите программу на исполнение. При запуске программы появится ошибка времени выполнения: **Индекс выходит за границы массива**. Она является сообщением о том, что значение переменной **i** (индекс массива **A**) вышло за допустимый предел, т.е. за значение константы **n**.
11. Формирование значений элементов массива путем ввода их с клавиатуры – достаточно утомительное занятие. Используйте для этих целей генератор случайных чисел - **Random**. Данная функция возвращает случайное число. Выполните команду **Файл – Новый** и составьте программу для заполнения массива случайными числами. Сохраните полученную программу в папке **Array1** под именем **Zad6.pas**.



```

Program Zad6;
Const n=10;
Type MyArray=Array[1..n] Of Integer;
Var
  A:MyArray;
  i:Integer;
Begin
  {Randomize;}
  WriteLn('Формирование значений элементов массива A');
  For i:=1 To n Do A[i]:=Random(21) {-10};
  WriteLn('Вывод');
  For i:=1 To n Do Write(A[i]:5);
End.

```

12. Запустите несколько раз программу. На экране мы видим одну и ту же последовательность чисел в диапазоне от 0 до 20.
13. Уберите фигурные скобки у -10 и снова запустите несколько раз программу.

`A[i]:=Random(21)-10;`

Последовательность чисел на экране одна и та же, но числа из интервала от -10 до 10. Объясните этот факт. Получите числа из интервала от -17 до 25.

14. Уберите фигурные скобки у процедуры **Randomize**. Повторите многократный запуск программы. Последовательности чисел на экране разные. В чем дело? Наш черный ящик, функция **Random**, начинает генерировать в первом случае числа (каким-то неизвестным нам образом) от фиксированного начального числа. Во втором случае эти начальные числа меняются от запуска к запуску (процедурой **Randomize**) и последовательности получаются разные.

## Задание 2.

Используя встроенный задачник системы программирования **PascalABC.NET**, выполните решение задач **Array4**, **Array7**.

**Методические указания по выполнению задания:**

1. Создайте новое окно, выполнив команду **Файл – Новый**.
2. Откройте встроенный задачник системы программирования **PascalABC.NET**, для этого выполните команду **Модули – Просмотреть задания**. В диалоговом окне **Просмотр учебных заданий** в поле группа заданий выберите **Array**, а в поле номер задания задайте необходимый номер задания и нажмите кнопку **Просмотр**.
3. Решите данную задачу, используя систему программирования **PascalABC.NET**, для этого перейдите в систему программирования **PascalABC** и выполните команду **Модули – Создать шаблон программы**. В окне **Загрузка учебного задания** в поле **Задание** введите **Array4 (Array7)** и нажмите кнопку **Загрузка**. После этого в системе программирования **PascalABC** загрузится шаблон программы, где указан модуль подключения задачника и название задачи, которую мы решаем.
4. Просмотрите еще раз решение задачи, для этого выполните команду **Программы – Выполнить** или нажмите соответствующую кнопку на панели инструментов.
5. В разделе описания переменных **var** выполните описание переменных в соответствии с условием задачи.
6. Составьте программу для решения задачи и выполните её тестирование. При необходимости исправьте ошибки. Чтобы задание считалось выполненным, запустите программу необходимое количество раз.
7. Сохраните созданные программы в папке **Array1**.

## Задание 3.

Используя систему программирования **PascalABC.NET**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **DiagramDesigner**. Созданные программу и её блок-схему сохранить в папку **Array1**, в названии файла укажите номер задачи.

**Задания:**

1. Вывести элементы массива на экран в обратном порядке.
2. Дан массив. Все его элементы: увеличить в 2 раза; уменьшить на число A; разделить на первый элемент.
3. Дан массив целых чисел. напечатать все четные элементы; все элементы, оканчивающиеся нулем.



#### Задание 4.

Проверьте файлы, которые были сохранены в папку **Array1** в процессе выполнения практической работы. Заархивируйте папку с помощью программы архиватора, установленной на ваш компьютер. Передайте полученный архив преподавателю на проверку, разместив его в общих папках.

### Практическая работа №9.

#### «Операции с элементами массивов, обмен элементами»

**Цель работы:** сформировать умения по использованию процедурного языка программирования для построения логически правильных и эффективных программ с использованием одномерных массивов.

**Оборудование, технические и программные средства:** персональный компьютер, среда программирования **Visual Studio**.

#### Задание 1.

В системе программирования **Visual Studio** составьте программу по заданию. Протестируйте работу программы. Выполните задания на модификацию созданной программы.

##### Методические указания по выполнению задания:

1. Запустите среду программирования **Visual Studio**.
2. В папке своей группы на диске создайте папку **Массивы1**, в дальнейшем все программы необходимо сохранять в эту папку.
3. Создайте новое консольное приложение с именем **Zad1**. В окне редактора наберите текст программы, которая находит сумму 30 целых чисел.

```
static int[] Input()
{
    int[] a = new int[30];
    for (int i = 0; i < 30; ++i)
    {
        Console.Write("a[{0}]= ", i);
        a[i] = int.Parse(Console.ReadLine());
    }
    return a;
}

static int Sum(int[] a)
{
    int sum = 0;
    foreach (int elem in a)
    {
        sum += elem;
    }
    return sum;
}

static void Main()
{
    int[] a = Input();
    Console.WriteLine("Сумма элементов массива={0}", Sum(a));
}
```

4. Проверьте правильность работы программы.
5. Создайте новое консольное приложение с именем **Zad2**. Измените ранее созданную программу так, чтобы определялась сумма элементов массива, кратных заданному числу.

Решение задачи изменится незначительным образом. Добавляется описание еще одной переменной для хранения значения числа, на кратность которому проверяются значения элементов массива.

Появляется оператор ввода заданного числа и изменяется оператор из тела цикла метода **Sum**:



```

static int Sum(int[] a)
{
    int sum = 0;
    Console.Write("Введите число k=");
    int k = int.Parse(Console.ReadLine());
    foreach (int elem in a)
    {
        if (elem%k==0)
        {
            sum += elem;
        }
    }
    return sum;
}

```

6. Создайте новое консольное приложение с именем **Zad3**. Измените программу так, чтобы определялось количество положительных и отрицательных элементов в данном массиве.

Суть основного изменения программы заключается во введении двух переменных (счетчиков – **pos**, **neg**) для хранения значений количества положительных и отрицательных элементов в массиве.

Внесите в программу следующие изменения:

```

static void Sum(int[] a)
{
    int pos = 0;
    int neg = 0;
    foreach (int elem in a)
    {
        if (elem>0)
        {
            pos++;
        }
        else
        {
            if (elem<0)
            {
                neg++;
            }
        }
    }
    Console.WriteLine("Кол-во положительных={0}", pos);
    Console.WriteLine("Кол-во отрицательных={0}", neg);
}

static void Main()
{
    int[] a = Input();
    Sum(a);
}

```

7. Создайте новое консольное приложение с именем **Zad4**. Измените программу так, чтобы в массив **В** записывались номера четных элементов массива **А**.

Введение массива **В** и работа с ним требует введения переменной **j** для обращения к элементам **В**. Суть решения заключается в просмотре элементов массива **А**, выявлении четных элементов и записи их номеров по текущему значению переменной **j** в массив **В**.

Текст программы будет иметь следующий вид:



```

static int[] Input()
{
    int[] a = new int[10];
    for (int i = 0; i < 10; ++i)
    {
        Console.Write("a[{0}]= ", i);
        a[i] = int.Parse(Console.ReadLine());
    }
    return a;
}
static void Print(int[] a, int j)
{
    for (int i = 0; i < j; ++i)
    {
        Console.Write("{0} ", a[i]);
    }
}

static void Main()
{
    int[] a = Input();
    int[] b;
    b = new int[10];
    int j = 0;
    for (int i = 0; i < 10; ++i)
    {
        if (a[i] % 2 == 0)
        {
            b[j] = i;
            j++;
        }
    }
    Print(b,j);
}

```

8. Создайте новое консольное приложение с именем **Zad5**. Составьте программу нахождения значения и номера всех отрицательных элементов массива.

Формирование значений элементов массива путем ввода их с клавиатуры – достаточно утомительное занятие. Используйте для этих целей генератор случайных чисел - **Random**.

```

static int[] Input()
{
    Random rnd = new Random();
    int[] a = new int[10];
    for (int i = 0; i < 10; ++i)
    {
        a[i] = 7-rnd.Next(10);
    }
    return a;
}

static void Main()
{
    int[] a = Input();
    Console.WriteLine("Вывод отрицательных элементов массива и их номеров");
    for (int i = 0; i < 10; ++i)
    {
        if (a[i]<0)
        {
            Console.WriteLine("A[{0}]=-{1}, i={0}", i, a[i]);
        }
    }
}

```

## Задание 2.

Используя систему программирования **Visual Studio**, выполнить индивидуальные задания. Для каждой задачи построить блок-схему алгоритма решения, используя инструменты **DiagramDesigner**. Созданные программу и её блок-схему сохранить в папку **Массивы1**, в названии файла укажите номер задачи.

### Задания:

1. Вывести элементы массива на экран в обратном порядке.
2. Дан массив. Все его элементы: увеличить в 2 раза; уменьшить на число А; разделить на первый элемент.
3. Дан массив целых чисел, напечатать все элементы массива, оканчивающиеся нулем.



## **5. МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ ПО ОРГАНИЗАЦИИ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ**

### **5.1. Самостоятельная работа студента на лекции**

После прослушивания лекции студент должен проработать и осмыслить полученный материал. Подготовка к самостоятельной работе над лекцией должна начинаться на самой лекции. Умение слушать, творчески воспринимать излагаемый материал – это необходимое условие для его понимания.

В процессе лекционного занятия студент должен выделять важные моменты, выводы, анализировать основные положения. Если при изложении материала преподавателем создана проблемная ситуация, пытаться предугадать дальнейший ход рассуждений. Это способствует лучшему усвоению материала лекции и облегчает запоминание отдельных выводов.

Недостаточно только слушать лекцию. Возможности памяти человека не универсальны. Как бы внимательно студент не слушал лекцию, большая часть информации вскоре после восприятия будет забыта.

Повторение и воспроизведение осуществляется при подготовке к практическим и лабораторным занятиям, контрольным.

Для более прочного усвоения знаний лекцию необходимо конспектировать. Конспект лекций должен быть в отдельной тетради. Не надо стремиться подробно слово в слово записывать всю лекцию. Конспектируйте только самое важное в рассматриваемом параграфе:

- формулировки определений и законов, выводы основных уравнений и формул,

- то, что старается выделить лектор, на чем акцентирует внимание студентов.

Старайтесь отфильтровывать и сжимать подаваемый материал. Более подробно записывайте основную информацию и кратко – дополнительную. Научитесь в процессе лекции разбивать текст на смысловые части и заменять их содержание короткими фразами и формулировками. Не нужно просить лектора несколько раз повторять одну и ту же фразу для того, чтобы успеть записать. По возможности записи ведите своими словами, своими формулировками.

Тетрадь для конспекта лекций нужно сделать удобной, практичной и полезной, ведь именно она является основным информативным источником при подготовке к различным отчетным занятиям, зачетам, экзаменам. Целесообразно отделить поля, где студент мог бы изложить свои мысли, вопросы, появившиеся в ходе лекции. Полезно одну из страниц оставлять свободной. Она потребуется потом, при самостоятельной подготовке. Сюда можно будет занести дополнительную информацию по данной теме, полученную из других источников.

Таким образом, на лекции студент должен совместить два момента:

- внимательно слушать лектора, прикладывая максимум усилий для



понимания излагаемого материала;

- одновременно вести его осмысленную запись.

## **5.2. Самостоятельная работа студента над лекцией**

Прослушанный материал лекции студент должен проработать. Насколько эффективно он это сделает, зависит и прочность усвоения знаний. Опыт показывает, что только многоразовая, планомерная и целенаправленная обработка лекционного материала обеспечивает его надежное закрепление в долговременной памяти человека.

Повторение нужно разнообразить. При первом повторении изучаются все параграфы и абзацы, при втором, возможно, будет достаточно рассмотреть только отдельные параграфы, а в дальнейшем лишь тему лекции.

Рекомендуется обучающимся составлять подробный конспект лекций. Особенно полезной эта работа оказывается в том случае, когда студенты знакомятся с теми вопросами, которые им еще необходимо как следует осмыслить. Осмысление и происходит во время описания материала своими словами, разъяснения его в первую очередь для себя. Естественно, что это конспектирование совершенно не то, что запись со слов лектора. Поэтому конспект, ведущийся студентами с целью осмысления и усвоения материала, получил название «свой собственный конспект» (ССК) ССК ведется на основе записей лекций, книг (вообще говоря, разных), консультаций преподавателей, бесед с товарищами и, конечно, в результате размышлений. Главная роль ССК заключается в том, что он помогает пониманию изучаемого предмета.

Правило 1. ССК нужно записывать своими словами, следовательно, лишь после того, как излагаемый в нём материал будет вам ясен.

Правило 2. Основой для составления ССК могут служить учебники (лучше, чтобы книг было несколько) и конспект лекций.

Правило 3. При составлении ССК следует придерживаться плана, который у вас должен иметься заранее, по крайней мере, для описываемой вами завершённой части курса.

Правило 4. При описании отдельного вопроса не обязательно точно придерживаться того порядка изложения, который был в вашем основном источнике (книге или конспекте лекций).

Правило 5. Составляя ССК, старайтесь в каждом более или менее законченном пункте выразить свое мнение по отношению к вопросам, помогающим осмыслению.

Правило 6. Приводя доказательство, описание, рассуждение, не оставляйте что-либо непонятым, записанным формально.

## **5.3. Работа с учебником**

При работе с учебником необходимо подобрать литературу, научиться правильно ее читать, вести записи. Для подбора литературы в библиотеке используются алфавитный и систематический каталоги.



Правильный подбор учебников рекомендуется преподавателем, читающим лекционный курс. Необходимая литература может быть также указана в методических разработках по данному курсу.

Изучая материал по учебнику, следует переходить к следующему вопросу только после правильного уяснения предыдущего, описывая на бумаге все выкладки и вычисления (в том числе те, которые в учебнике опущены или на лекции даны для самостоятельного вывода).

При изучении любой дисциплины большую и важную роль играет самостоятельная индивидуальная работа.

#### **5.4. Практические занятия**

Для того чтобы практические занятия приносили максимальную пользу, необходимо помнить, что упражнение и решение задач проводятся по вычитанному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях студент не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если студент видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

#### **5.5. Подготовка доклада**

Доклад – публичное сообщение на определенную тему, способствующее формированию навыков исследовательской работы, расширяющее познавательный интерес.

Работа над докладом состоит из следующих этапов:

- составление плана работы;
- систематизации полученных сведений;



- составление выводов и обобщений.

Доклад может быть представлен в устной и письменной форме.

Письменный доклад – это запись устного сообщения по какой-либо теме объёмом от пяти до пятнадцати страниц. В таком докладе не обязательно:

- выделять структурные элементы работы в виде плана;
- выделять заголовки внутри текста;
- ссылаться на использованную литературу по ходу текста.

Но обязательно следует приводить список всех используемых источников в конце работы. При подготовке доклада целесообразно соблюдать следующий порядок работы:

- подобрать литературу по изучаемой теме, познакомиться с её содержанием;
- пользуясь закладками, отметить наиболее существенные места или сделать выписки;
- составить план доклада;
- используя рекомендации по составлению тематического конспекта и составленный план, написать доклад, в заключение которого обязательно выразить своё отношение к излагаемой теме и её содержанию;
- прочитать текст и редактировать его;
- оформить в соответствии с требованиями к оформлению докладов.

Регламент устного публичного выступления – не более 10 минут.

Работу по подготовке устного выступления можно разделить на два основных этапа:

- докоммуникативный этап (подготовка выступления);
- коммуникативный этап (взаимодействие с аудиторией).

Работа по подготовке устного выступления начинается с формулировки темы. Лучше всего тему сформулировать таким образом, чтобы ее первое слово обозначало наименование полученного в ходе выполнения проекта научного результата (например, «Технология изготовления...», «Модель развития...», «Система управления...» и пр.).

Само выступление должно состоять из трех частей – вступления (10-15% общего времени), основной части (60-70%) и заключения (20-25%).

Вступление включает в себя представление авторов, название сообщения, расшифровку подзаголовка с целью точного определения содержания выступления, четкое определение стержневой идеи. Стержневая идея проекта понимается как основной тезис, ключевое положение.

План развития основной части должен быть ясным. Должно быть отобрано оптимальное количество фактов и необходимых примеров.

В заключении необходимо сформулировать выводы, которые следуют из основной идеи (идей) выступления. Правильно построенное заключение способствует хорошему впечатлению от выступления в целом. В заключении имеет смысл повторить стержневую идею и, кроме того, вновь (в кратком виде) вернуться к тем моментам основной части, которые вызвали интерес слушателей. Закончить выступление можно решительным заявлением.



При подготовке к выступлению необходимо выбрать способ выступления: устное изложение с опорой на конспект (опорой могут также служить заранее подготовленные слайды) или чтение подготовленного текста.

После выступления нужно быть готовым к ответам на возникшие у аудитории вопросы.

### **5.6. Подготовка презентации**

Компьютерную презентацию, сопровождающую выступление докладчика, удобнее всего подготовить в программе MS PowerPoint. Презентация как документ представляет собой последовательность сменяющих друг друга слайдов - то есть электронных страничек, занимающих весь экран монитора. Чаще всего демонстрация презентации проецируется на большом экране, реже – раздается собравшимся как печатный материал. Количество слайдов адекватно содержанию и продолжительности выступления (например, для 5-минутного выступления рекомендуется использовать не более 10 слайдов).

На первом слайде обязательно представляется тема выступления и сведения об авторах. Следующие слайды можно подготовить, используя две различные стратегии их подготовки:

**1 стратегия:** на слайды выносятся опорный конспект выступления и ключевые слова с тем, чтобы пользоваться ими как планом для выступления. В этом случае к слайдам предъявляются следующие требования:

- объем текста на слайде – не больше 7 строк;
- маркированный/нумерованный список содержит не более 7 элементов;
- отсутствуют знаки пунктуации в конце строк в маркированных и нумерованных списках;
- значимая информация выделяется с помощью цвета, кегля, эффектов анимации.

Особо внимательно необходимо проверить текст на отсутствие ошибок и опечаток. Основная ошибка при выборе данной стратегии состоит в том, что выступающие заменяют свою речь чтением текста со слайдов.

**2 стратегия:** на слайды помещается фактический материал (таблицы, графики, фотографии и пр.), который является уместным и достаточным средством наглядности, помогает в раскрытии стержневой идеи выступления. В этом случае к слайдам предъявляются следующие требования:

- выбранные средства визуализации информации (таблицы, схемы, графики и т. д.) соответствуют содержанию;
- использованы иллюстрации хорошего качества (высокого разрешения), с четким изображением (как правило, никто из присутствующих не заинтересован вчитываться в текст на ваших слайдах и всматриваться в мелкие иллюстрации).

Максимальное количество графической информации на одном слайде – 2 рисунка (фотографии, схемы и т.д.) с текстовыми комментариями (не более 2



строк к каждому). Наиболее важная информация должна располагаться в центре экрана.

Особо тщательно необходимо отнестись к оформлению презентации. Для всех слайдов презентации по возможности необходимо использовать один и тот же шаблон оформления, кегль – для заголовков - не меньше 24 пунктов, для информации - для информации не менее 18. В презентациях не принято ставить переносы в словах.

### **5.7. Консультации**

Разъяснение является основным содержанием данной формы занятий, наиболее сложных вопросов изучаемого программного материала. Цель – максимальное приближение обучения к практическим интересам с учетом имеющейся информации и является результативным материалом закрепления знаний.

Групповая консультация проводится в следующих случаях:

- когда необходимо подробно рассмотреть практические вопросы, которые были недостаточно освещены или совсем не освещены в процессе лекции или практического занятия;
- с целью оказания помощи в самостоятельной работе (написание рефератов, выполнение курсовых работ, сдача экзаменов, подготовка конференций);
- если обучающиеся самостоятельно изучают нормативный, справочный материал, инструкции, положения.

Проведение групповой консультации предполагает наличие у студентов заранее подготовленных вопросов. Список вопросов формируется в процессе изучения дисциплины. Желательно конспектирование вопросов, задаваемых другими студентами группы и ответов на них (выводов).

### **5.8. Подготовка к экзамену**

Подготовка к экзамену способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к экзамену, студент ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене студент демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Требования к организации подготовки к экзаменам те же, что и при занятиях в течение семестра, но соблюдаться они должны более строго. Во-первых, очень важно соблюдение режима дня; сон не менее 8 часов в сутки, занятия заканчиваются не позднее, чем за 2-3 часа до сна.

В перерывах между занятиями рекомендуются прогулки на свежем воздухе, неустойчивые занятия спортом.

Во-вторых, наличие хороших собственных конспектов лекций. Даже в том случае, если была пропущена какая-либо лекция, необходимо во время ее



восстановить (переписать у товарища), обдумать, снять возникшие вопросы для того, чтобы запоминание материала было осознанным.

В-третьих, при подготовке к экзаменам у студента должен быть хороший учебник или конспект литературы, прочитанной по указанию преподавателя в течение семестра.

Вначале следует просмотреть весь материал по сдаваемой дисциплине, отметить для себя трудные вопросы. Обязательно в них разобраться. В заключение еще раз целесообразно повторить основные положения, используя при этом листы опорных сигналов.

Систематическая подготовка к занятиям в течение семестра позволит использовать время экзаменационной сессии для систематизации знаний.