**Муниципальное бюджетное общеобразовательное учреждение:**

**Лагутнинская средняя общеобразовательная школа**

**ПРОГРАММА**

**элективного курса «С++. Программирование на языке высокого уровня»**

Класс: 10 – 11

Автор: Прутко И.В.

Оглавление

[ПОЯСНИТЕЛЬНАЯ ЗАПИСКА 3](#_Toc448328046)

[ТРЕБОВАНИЯ К ЗНАНИЯМ И УМЕНИЯМ УЧАЩИХСЯ 6](#_Toc448328047)

[ТЕМАТИЧЕСКИЙ ПЛАН 7](#_Toc448328048)

[Содержание ЭЛЕКТИВНОГО КУРСА 8](#_Toc448328049)

[Перечень практических работ 14](#_Toc448328050)

[Практическая работа №1 15](#_Toc448328051)

[Практическая работа №2 17](#_Toc448328052)

[Практическая работа №3 20](#_Toc448328053)

[Практическая работа №4 22](#_Toc448328054)

[Практическая работа №5 25](#_Toc448328055)

[Практическая работа №6 27](#_Toc448328056)

[Практическая работа №7 29](#_Toc448328057)

[Практическая работа №8 31](#_Toc448328058)

[Практическая работа №9 33](#_Toc448328059)

[Практическая работа №10 35](#_Toc448328060)

[Практическая работа №11 37](#_Toc448328061)

[Практическая работа №12 39](#_Toc448328062)

[Практическая работа №13 41](#_Toc448328063)

[Практическая работа №14 43](#_Toc448328064)

[Практическая работа №15 46](#_Toc448328065)

[ВОПРОСЫ К ЗАЧЕТУ 48](#_Toc448328066)

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

В современном мире произошло существенное изменение взглядов на роль и место компьютерных технологий в жизни человека. Персональный компьютер все чаще используется как средство обучения, бумажные книги постепенно вытесняются электронными учебниками, технологии дистанционного обучения позволяют получать новые знания прямо на своем рабочем месте. Все это приводит к тому, что в школьном курсе информатики основное количество часов отводится на изучение прикладного программного обеспечения, а такие разделы как «Алгоритмизация» и «Программирование» либо полностью исключаются, либо изучаются поверхностно. В результате современный школьник приобретает лишь навыки работы с программным обеспечением компьютера, которые зачастую теоретически не подкреплены. Суть многих операций понимается ими поверхностно, главным становится технологический навык, а не владение методом.

Велика роль изучения программирования и для развития мышления школьников, формирования многих общеучебных, общеинтеллектуальных умений и навыков.

Кроме того, задания из разделов «Алгоритмизация» и «Программирование» включаются также в предметные олимпиады и КИМы ЕГЭ по информатике, такие задачи рассматриваются при изучении курса информатики в ВУЗах и ССУЗах, поэтому при малом количестве времени, отведенном на изучение этих тем, элективный курс «С++. Программирование на языке высокого уровня» поможет изучить основы алгоритмизации и программирования в более полном объеме.

Программа элективного курса «С++. Программирование на языке высокого уровня» является авторской. Данный курс является предметом по выбору для учащихся 10-11 классов и рассчитан на 68 часов (1 час в неделю).

**Цели курса**:

* формирование интереса к изучению профессии, связанной с программированием;
* формирование знаний, умений и навыков в области программирования на языке С++;

**Задачи курса**:

* приобретение знаний и навыков составления алгоритмов;
* развитие алгоритмического мышления;
* обучение структурному программированию;
* освоение методов решения задач, реализуемых на языке С++;
* формирование навыков грамотной разработки программ;
* углубление знаний, умений и навыков решения задач по программированию и алгоритмизации.

**Состав учебно-методического комплекта**:

* С/С++. Программирование на языке высокого уровня / Т.А. Павловская. – СПб.: Питер, 2002. – 464 с.: ил.
* Язык программирования С++ / Бьерн Страуструп. – Бином, Невский Диалект , 2008. – 1104.

**Контроль знаний и умений**. Текущий контроль уровня усвоения материала осуществляется по результатам выполнения учащимися практикумов по каждому разделу курса.

**Формы организации учебного процесса**. Учебный процесс организуется в виде лекционно-семинарских занятий. Формы учебного процесса:

* **урочная форма** – учитель объясняет новый материал (лекции), консультирует учащихся в процессе решения задач, учащиеся защищают практикумы по решению задач;
* **внеурочная форма** – учащиеся после занятий самостоятельно выполняют компьютерные практикумы.

**Лекции**. Представление учебного материала в форме лекций с демонстрацией презентаций.

**Практикум**. Основной формой проведения занятий являются практикумы по написанию программ на С++. Организация личностно-ориентированных практикумов по решению задач, личностно-ориентированного контроля – это как раз то, что необходимо учащемуся для его уверенности, успешности в очень сложном разделе информатики. Эти две формы работы предполагают следующее:

* Каждому ученику подбираются индивидуальные задачи.
* Подбор задач для каждого ученика необходимо выполнять исходя из их умственных способностей и психологического настроя к программированию.
* Задачи каждому ученику выдаются адресно. Каждый ученик на разных практикумах имеет разный вариант (сегодня первый, в следующий раз девятый и т.д.) – это тоже важный момент, ориентированный на личность учащегося.
* Задачи для каждого ученика посильные, т.е. он заведомо уверен в своем успехе.

**Семинар**. После завершения практикума ученики защищают свои решения на семинарах перед другими учениками, делится новыми способами решения. Принимают участие в дискуссии по поводу решения задач, предлагают другие пути их решения. Отвечают на возникшие вопросы в ходе обсуждения.

**Итоговая аттестация** по окончанию курса проводится в форме зачета.

# ТРЕБОВАНИЯ К ЗНАНИЯМ И УМЕНИЯМ УЧАЩИХСЯ

В результате изучения курса обучающиеся:

***должны знать***:

* структуру программы на языке С++;
* основные типы констант;
* основные операции в языке С++;
* основные операторы;
* способы создания и работы с массивами;
* определение и базовую структуру функции в программе;
* этапы создания программ.

***должны уметь***:

* применять различные операции языка С++ при написании программы;
* составлять выражения;
* применять для решения задач различные виды операторов и циклов;
* работать с одномерными и многомерными массивами;
* работать с функциями.

# ТЕМАТИЧЕСКИЙ ПЛАН

**элективного курса** «С++. Программирование на языке высокого уровня»

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование **разделов и тем** | **Количество аудиторных часов** | | |
| **Всего** | **В том числе** | |
| **Лабораторные работы** | **Практические работы** |
| *1* | *2* | *3* | *4* |
| Введение | **1** |  |  |
| **Раздел 1. Базовые средства языка С++** | **33** |  | **20** |
| **Тема 1.1.** Состав языка С++ | 1 |  |  |
| **Тема 1.2.** Типы данных С++ | 2 |  |  |
| **Тема 1.3.** Переменные и выражения | 7 |  | 4 |
| **Тема 1.4.** Базовые конструкции структурного программирования | 11 |  | 8 |
| **Тема 1.5.** Указатели и массивы | 12 |  | 8 |
| **Раздел 2. Модульное программирование** | **22** |  | **10** |
| **Тема 2.1.** Функции | 18 |  | 10 |
| **Тема 2.2.** Директивы препроцессора | 2 |  |  |
| **Тема 2.3.** Области действия идентификаторов | 2 |  |  |
| **Раздел 3. Технология создания программ** | **12** |  |  |
| **Тема 3.1.** Кодирование и документирование программы | 3 |  |  |
| **Тема 3.2.** Проектирование и тестирование программы | 3 |  |  |
| **Тема 3.3.** Динамические структуры данных | 6 |  |  |
| **Всего часов:** | **68** |  | **30** |

# Содержание ЭЛЕКТИВНОГО КУРСА

«С++. Программирование на языке высокого уровня»

**Введение**

*Обучающийся должен иметь представление*:

* о разновидностях системных языков программирования;
* о роли языка С++ в программировании.

Системные языки программирования: разновидности, история развития, языки высокого и низкого уровня.

Язык С++: роль языка С++ в программировании, возможности языка.

**Раздел 1. Базовые средства языка С++**

**Тема 1.1 Состав языка С++**

*Обучающийся должен знать*:

* состав алгоритмического языка;
* алфавит языка С++;
* знаки операций;
* типы констант.

Состав языка: лексемы, выражения, операторы, машинный код, препроцессор, директива, компилятор, объектный модель, компоновщик, исполняемый модель.

Алфавит языка: прописные и строчные латинские буквы, знак подчеркивания, арабские цифры, пробельные символы; лексемы языка: идентификаторы, ключевые слова, знаки операций, константы, разделители.

Константы, типы констант: целая, вещественная, символьная, строковая. Формат констант. Комментарии.

**Тема 1.2. Типы данных С++**

*Обучающийся должен знать*:

* основные типы данных;
* спецификаторы типа;
* структуру программы.

Типы данных С++: целый, символьный, расширенный символьный, логический, вещественный, вещественный с двойной точностью.

Спецификаторы типа: короткий, длинный, знаковый, беззнаковый.

Структура программы: функция, структура функции, описание, директивы препропроцессора, функция main(), функции возвращающие и не возвращающие значение.

**Тема 1.3. Переменные и выражения**

*Обучающийся должен знать*:

* назначение переменной и выражения;
* область действия и виды переменных;
* основные операции в языке С++;
* правила составления выражений.

*Обучающийся должен уметь*:

* описывать различные виды переменных;
* разработать простейшую функцию для работы с переменными;
* применять различные операции языка С++ при написании программы;
* составлять выражения.

Переменные: определение, назначение, способы описания, константы и их инициализация, область действия и виды переменных, время жизни переменных, спецификаторы: auto, extern, static, register.

Операции в языке С++: унарные, бинарные и тернарные операции.

Правила составления выражений: операнды, знаки операций, скобки, приоритеты, преобразование двух типов.

**Практическая работа №1. Создание программы для работы с переменными.**

**Практическая работа №2. Создание программы для вычисления значений функции y=y(x).**

**Тема 1.4. Базовые конструкции структурного программирования**

*Обучающийся должен знать*:

* понятие базовой конструкции;
* основные виды конструкций;
* синтаксис операторов ветвления;
* синтаксис операторов цикла;
* синтаксис операторов передачи управления.

*Обучающийся должен уметь*:

* строить конструкции;
* применять для решения задач различные виды операторов и циклов.

Базовые конструкции структурного программирования: понятие базовой конструкции. Основные виды конструкций: следование, ветвление, цикл.

Операторы ветвления: условный оператор if, оператор switch, синтаксис.

Операторы цикла: цикл с предусловием, цикл с постусловием, цикл с параметром, синтаксис.

Операторы передачи управления: оператор goto, оператор break, синтаксис.

**Практическая работа №3. Работа с конструкцией следования.**

**Практическая работа №4. Работа с операторами ветвления.**

**Практическая работа №5. Работа с операторами цикла.**

**Практическая работа №6. Работа с операторами передачи управления.**

**Тема 1.5. Указатели и массивы**

*Обучающийся должен знать*:

* понятие и назначение указателя;
* виды указателей;
* способы инициализации указателей;
* основные операции с указателями;
* понятие и назначение ссылки
* понятие и назначение массива;
* основные виды массивов;
* способы создания массивов;
* способы работы с элементами массивов;
* способы работы со строками.

*Обучающийся должен уметь*:

* работать с одномерными и многомерными массивами;

Указатели: понятие, назначение, виды, способы инициализации, операции с указателями.

Ссылки: понятие и назначение.

Массивы: понятие, назначение, описание, нумерация элементов, размерность, доступ к элементу, динамические массивы, многомерные массивы, работа с элементами массива.

Строки: понятие, способы работы со строками.

**Практическая работа №7. Создание и работа со статическими одномерными массивами.**

**Практическая работа №8. Создание и работа со статическими двумерными массивами.**

**Практическая работа №9. Сортировка одномерных массивов.**

**Практическая работа №10. Поиск минимального значения функции на заданном промежутке.**

**Раздел 2. Модульное программирование**

**Тема 2.1. Функции**

*Обучающийся должен знать*:

* понятие и назначение функций;
* определение и базовую структуру функции в программе;
* способы передачи параметров в функцию;
* способы передачи массивов в функцию в качестве параметров;
* структуру рекурсивной функции;
* форматы функции main().

*Обучающийся должен уметь*:

* работать с функциями;
* работать с функциями стандартной библиотеки.

Функции: понятие, назначение, определение и базовая структура функции в программе, тип возвращаемого значения, использование глобальных переменных, механизм возврата из функции.

Параметры функции: формальные параметры, фактические параметры.

Способы передачи параметров в функцию: по адресу и по значению.

Передача массивов в качестве параметров, передача одномерных и многомерных массивов в качестве параметров.

Передача имен функций в качестве параметров.

Параметры со значением по умолчанию.

Функции с переменным числом параметров.

Рекурсивные функции: понятие, виды рекурсий, структура рекурсивной функции, область применения.

Функция main(): форматы и параметры функции main().

Функции стандартной библиотеки: функции ввода/вывода, функции работы со строками и символами.

**Практическая работа №11. Работа с функциями.**

**Практическая работа №12. Передача массивов в функцию в качестве параметров.**

**Практическая работа №13. Работа с рекурсивными функциями.**

**Практическая работа №14. Работа с файлами.**

**Практическая работа №15. Работа со строками и символами.**

**Тема 2.2. Директивы препроцессора**

*Обучающийся должен знать*:

* понятие препроцессора;
* директивы;

Препроцессор: понятие.

Директивы препроцессора: #include, #define, #undef, директивы условной компиляции.

**Тема 2.3. Области действия идентификаторов**

*Обучающийся должен знать*:

* виды областей действия идентификаторов;
* способы ограничения области действия функции;
* назначение поименованной области.

Область действия идентификаторов, виды областей действия: блок, файл, функция, прототип функции, класс, поименованная область.

Внешние объявления: способы ограничения области действия функции.

Поименованные области: назначение, объявление поименованной области, пространства имен стандартной библиотеки.

**Раздел 3. Технология создания программ**

**Тема 3.1. Кодирование и документирование программы**

*Обучающийся должен знать*:

* особенности и рекомендации для создания программ;

Особенности и рекомендации для создания программы: подбор имен переменных, инициализация переменных, использование локальных и глобальных переменных, передача информации в функцию, использование константных ссылок в функциях, использование чисел в явном виде, лишние проверки условий, проверка на неравенство нулю, использование ветвления, циклов, проверка кодов возврата ошибок, использование сообщений об ошибке, комментариев, вложенных блоков.

**Тема 3.2. Проектирование и тестирование программы**

*Обучающийся должен знать*:

* этапы создания программ.

Этапы создания программ.

Постановка задачи: техническое задание, внешняя спецификация программы, ее состав.

Разработка внутренних структур данных: статические и динамические структуры данных.

Проектирование, структурное программирование: технология нисходящего проектирования, спецификация интерфейсов, учет будущих модификаций программы.

Структурное программирование: определение, установка «заглушек», этапы проектирования и программирования.

Нисходящее тестирование: написание набора тестов, понятие тестирования, отладки, проверка ветвей алгоритма, граничных условий, реакция программы на ошибочные исходные данные.

**Тема 3.3. Динамические структуры данных**

*Обучающийся должен знать*:

* понятие динамической структуры данных;
* виды динамических структур данных;

Динамические структуры данных: назначение.

Виды динамических структур данных.

Элемент динамической структуры данных, его состав.

Линейные списки: определение, виды, ключ, основные операции над списком.

Стеки: определение, назначение, операции над стеками.

Очереди: определение, назначение, операции над очередью.

Бинарные деревья: определение, структура, операции над бинарными деревьями.

**Перечень практических работ**

элективного курса «С++. Программирование на языке высокого уровня»

|  |  |  |  |
| --- | --- | --- | --- |
| **№**  **практической работы** | **№ раздела (темы)** | **Наименование практической работы** | **Кол-во часов** |
| 1 | Тема 1.3. | Создание программы для работы с переменными. | 2 |
| 2 | Тема 1.3. | Создание программы для вычисления значений функции y=y(x). | 2 |
| 3 | Тема 1.4. | Работа с конструкцией следования. | 2 |
| 4 | Тема 1.4. | Работа с операторами ветвления. | 2 |
| 5 | Тема 1.4. | Работа с операторами цикла. | 2 |
| 6 | Тема 1.4. | Работа с операторами передачи управления | 2 |
| 7 | Тема 1.5. | Создание и работа со статическими одномерными массивами. | 2 |
| 8 | Тема 1.5. | Создание и работа со статическими двумерными массивами. | 2 |
| 9 | Тема 1.5. | Сортировка одномерных массивов. | 2 |
| 10 | Тема 1.5. | Поиск минимального значения функции на заданном промежутке. | 2 |
| 11 | Тема 2.1. | Работа с функциями. | 2 |
| 12 | Тема 2.1. | Передача массивов в функцию в качестве параметров. | 2 |
| 13 | Тема 2.1. | Работа с рекурсивными функциями. | 2 |
| 14 | Тема 2.1. | Работа с файлами. | 2 |
| 15 | Тема 2.1. | Работа со строками и символами. | 2 |
| **Итого** | | | **30** |

## Практическая работа №1

**«Создание программы для работы с переменными»**

**Цель**: освоить работу с переменными.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

*Переменная* — это именованная область памяти, в которой хранятся данные определенного типа. У переменной есть имя и значение. Имя служит для обращения к области памяти, в которой хранится значение. Во время выполнения программы значение переменной можно изменять. Перед использованием любая переменная должна быть описана.

Пример описания целой переменной с именем *a* и вещественной переменной *x*:

*int a*; *float x*;

Общий вид оператора описания переменных:

[класс памяти] [*const*] тип имя [инициализатор];

Константа должна быть инициализирована при объявлении. В одном операторе можно описать несколько переменных одного типа, разделяя их запятыми.

Например:

*short int a* = 1;

*const char C* = '*C*';

Если тип инициализирующего значения не совпадает с типом переменной, выполняются преобразования типа.

Класс памяти определяет время жизни и область видимости программного объекта. Для задания класса памяти используются следующие спецификаторы:

auto — автоматическая переменная.

extern — означает, что переменная определяется в другом месте программы.

static — статическая переменная. Время жизни такой переменной — постоянное.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**I вариант**

Написать программу, которая вычисляет сумму и произведение трех вещественных чисел, которые вводятся из клавиатуры. Результаты вычислений, а также сумму результатов вывести на экран.

**II вариант**

Написать программу, которая вычисляет разность и произведение четырех вещественных чисел, которые вводятся из клавиатуры. Результаты вычислений, а также сумму результатов вывести на экран.

**Контрольные вопросы**

1. Что называется переменной?
2. Напишите пример описания вещественной переменной.
3. Каков общий вид оператора описания переменных?
4. Какие спецификаторы используются для задания класса памяти?

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №2**

**«Создание программы для вычисления значений функции y = y(x)»**

**Цель**: освоить работу с переменными и операциями.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

В соответствии с количеством операндов, которые используются в операциях, различают унарные (один операнд), бинарные (два операнда) и тернарную (три операнда) операции.

Ниже перечислены основные операции языка С++.

|  |  |
| --- | --- |
| **Операция** | **Краткое описание** |
| **Унарные операции** | |
| ++ | Увеличение на 1 |
| -- | Уменьшение на 1 |
| sizeof | Размер |
| ~ | Поразрядное отрицание |
| ! | Логическое отрицание |
| - | Арифметическое отрицание |
| + | Унарный плюс |
| & | Взятие адреса |
| \* | Разадресация |
| new | Выделение памяти |
| delete | Освобождение памяти |
| (type) | Преобразование типа |
| **Бинарные и тернарные операции** | |
| \*  /  % | Умножение  Деление  Остаток от деления |
| +  - | Сложение  Вычитание |
| <<  >> | Сдвиг влево  Сдвиг вправо |
| <  <=  >  >= | Меньше  Меньше или равно  Больше  Больше или равно |
| ==  != | Равно  Не равно |
| & | Поразрядная конъюнкция (И) |
| ^ | Поразрядное исключающее ИЛИ |
| | | Поразрядная дизъюнкция (ИЛИ) |
| && | Логическое И |
| || | Логическое ИЛИ |
| ? :  =  \*=  /=  %=  +=  -=  <<=  >>=  &=  |=  ^= | Условная операция (тернарная)  Присваивание  Умножение с присваиванием  Деление с присваиванием  Остаток от деления с присваиванием  Сложение с присваиванием  Вычитание с присваиванием  Сдвиг влево с присваиванием  Сдвиг вправо с присваиванием  Поразрядное И с присваиванием  Поразрядное ИЛИ с присваиванием  Поразрядное исключающее ИЛИ с присваиванием |
| . | Последовательное вычисление |

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Дана функция . Написать программу для вычисления значений функции. Степени представить в виде произведения. Вычислить значения функции в десяти различных точках, значения координат *х* вводятся с клавиатуры. Результаты вычислений должны выводиться в консоль в формате: значение координаты *х*; значение координаты *y*.

**Вариант I**

Дана функция . Написать программу для вычисления значений функции. Степени представить в виде произведения. Вычислить значения функции в двенадцати различных точках, значения координат *х* вводятся с клавиатуры. Результаты вычислений должны выводиться в консоль в формате: значение координаты *х*; значение координаты *y*.

**Контрольные вопросы**

1. Что называется операцией?
2. Какие типы операций в С++ вы знаете?
3. Какие формы записи увеличения и уменьшения на 1 вы знаете, в чем их различия?
4. Напишите операцию, которая находит остаток от деления *х/y*.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

## Практическая работа №3

**«Работа с конструкцией следования»**

**Цель**: освоить работу с конструкциями следования.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Программу для решения задачи любой сложности можно составить только из трех структур, называемых следованием, ветвлением и циклом.

Следование, ветвление и цикл называют базовыми конструкциями структурного программирования. *Следованием* называется конструкция, представляющая собой последовательное выполнение двух или более операторов (простых или составных). *Ветвление* задает выполнение либо одного, либо другого оператора в зависимости от выполнения какого-либо условия. *Цикл* задает многократное выполнение оператора. Особенностью базовых конструкций является то, что любая из них имеет только один вход и один выход, поэтому конструкции могут вкладываться друг в друга произвольным образом, например, цикл может содержать следование из двух ветвлений, каждое из которых включает вложенные циклы.

Целью использования базовых конструкций является получение программы простой структуры. Такую программу легко читать, отлаживать и при необходимости вносить в нее изменения. Структурное программирование часто называют «программированием без оператора goto», потому как частое использование оператора передачи управления в произвольные точки программы затрудняет прослеживание логики ее работы.

В языке С++ есть три вида циклов и два вида ветвления. Они введены для удобства программирования, и в каждом случае надо выбирать наиболее подходящие средства. Главное, о чем нужно помнить даже при написании самых простых программ – они должны состоять из четкой последовательности блоков строго определенной конфигурации.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Разработать программу для вычисления значений функции:

,

где

,

.

Входными данными являются значения *z*, они вводятся с клавиатуры, выходными – значения *y*.

Вычислите значение функции в семи различных точках, результаты выведите на экран в формате: Значение параметра *z*; Значение *y*.

**Вариант II**

Разработать программу для вычисления значений функции:

,

где

,

.

Входными данными являются значения *z*, они вводятся с клавиатуры, выходными – значения *y*.

Вычислите значение функции в семи различных точках, результаты выведите на экран в формате: Значение параметра *z*; Значение *y*.

**Контрольные вопросы**

1. Что называется базовыми конструкциями?
2. Что называется следованием?
3. Что называется ветвлением?
4. Что называется циклом?
5. Нарисуйте схемы следования, ветвления и цикла, объясните суть их работы.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

## Практическая работа №4

**«Работа с операторами ветвления»**

**Цель**: освоить работу с операторами ветвления.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

**Условный оператор if.**

Условный оператор if используется для разветвления процесса вычислений на два направления. Формат оператора:

if (выражение) оператор\_1; [else оператор\_2;]

Сначала вычисляется выражение, которое может иметь арифметический тип или тип указателя. Если оно имеет значение true, выполняется первый оператор, иначе – второй. После этого управление передается на оператор, следующий за условным.

Одна из ветвей может отсутствовать, логичнее опускать вторую ветвь вместе с ключевым словом else. Если в какой-либо ветви требуется выполнить несколько операторов, их необходимо заключить в блок, иначе компилятор не сможет понять, где заканчивается блок. Блок может содержать любые операторы, в том числе описания и другие условные операторы. Необходимо учитывать, что переменная, описанная в блоке, вне блока не существует.

Пример:

if (a > 0) b = 1;

else b = 2;

Если требуется проверить несколько условий, их объединяют знаками логических операций

Пример:

if(a > 0 && b < 0)

{

c = a + b;

d = a + 2 \* d;

}

else

{

c = a – b;

d = a – 2 \* b;

}

**Оператора switch**

Оператор switch (переключатель) предназначен для разветвления процесса вычислений на несколько направлений. Формат оператора:

switch (выражение) {

case константное\_выражение\_1: [список\_операторов\_1]

case константное\_выражение\_2: [список\_операторов\_2]

....

[default: операторы ]

}

Выполнение оператора начинается с вычисления выражения (оно должно быть целочисленным), а затем управление передается первому оператору из списка, помеченного константным выражением, значение которого совпало с вычисленным. После этого, если выход из переключателя явно не указан, последовательно выполняются все остальные ветви.

Выход из переключателя обычно выполняется с помощью операторов break или return. Оператор breаk выполняет выход из самого внутреннего из объемлющих его операторов switch, for, while и do. Оператор return выполняет выход их функций, в теле которой он записан.

Все константные выражения должны иметь разные значения, но быть одного и того же целостного типа. Несколько меток могут следовать подряд. Если совпадения не произошло, выполняются операторы, расположеные после слова default (а при его отсутствии управление передается следующему switch оператору).

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Производится выстрел по квадратной мишени, координаты левого нижнего угла равны (0, 0), а правого верхнего угла – (3, 10). Напишите программу, которая определяет количество очков за 10 выстрелов, если за каждое верное попадание начисляется одно очко. Результат выведите на экран в формате: Количество выстрелов равно \_\_\_, количество очков равно \_\_\_. Координаты выстрелов вводятся с клавиатуры.

**Вариант II**

Производится выстрел по квадратной мишени, координаты левого нижнего угла равны (-2, -3), а правого верхнего угла – (30, 20). Напишите программу, которая определяет количество очков за 10 выстрелов, если за каждое верное попадание начисляется одно очко. Результат выведите на экран в формате: Количество выстрелов равно \_\_\_, количество очков равно \_\_\_. Координаты выстрелов вводятся с клавиатуры.

**Контрольные вопросы**

1. Каково назначение условного оператора?
2. Напишите формат оператора if.
3. Опишите принцип работы оператора if.
4. Каково назначение оператора switch?
5. Напишите формат оператора switch.
6. Опишите принцип работы оператора switch.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №5**

**«Работа с операторами цикла»**

**Цель**: освоить работу с операторами цикла.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Операторы цикла используются для организации многократно повторяющихся вычислений. Любой цикл состоит из тела цикла, то есть тех операторов, которые выполняются несколько раз, начальных установок, модификации параметра цикла и проверки условия продолжения и выполнения цикла.

Один проход цикла называется итерацией. Проверка условия выполняется на каждой итерации, либо до тела цикла (того говорят о цикле с предусловием), либо после тела цикла (цикл с постусловием). Разница между ними состоит в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется, надо ли его выполнять еще раз. Проверка необходимости выполнения цикла с предусловием делается до тела цикла, поэтому возможно, что он не выполнится ни разу.

Переменные, изменяющиеся в теле цикла и используемые при проверке условия продолжения, называются параметрами цикла. Целочисленные параметры цикла, изменяющиеся с постоянным шагом на каждой итерации, называются счетчиками.

Начальные установки могут явно не присутствовать в программе, их смысл состоит в том, чтобы до входа в цикл задать значения переменным, которые в нем используются.

Цикл завершается, если условие его продолжения не выполняется. Возможно принудительное завершение как текущей итерации, так и цикла в целом. Для этого служат операторы break, continue, return и goto.

В С++ есть три разных оператора цикла – while, do while и for.

Цикл с предусловием имеет вид:

while (выражение) оператор

Цикл с постусловием имеет вид:

do оператор while выражение;

Цикл с параметром имеет следующий формат:

for (инициализация; выражение; модификации) оператор;

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

Напишите программу для вычисления квадратного корня вещественного аргумента *x* с заданной точностью *Eps* по итерационной формуле: , где – предыдущее приближение к корню (в начале вычислений выбирается произвольно). Для вычисления абсолютной величины используется стандатная функция *fabs*(), объявление которой находится в заголовочном файле *math.h*. Входными данными является число *x* и величина точности *Eps* (вводятся с клавиатуры), выходные данные – вычисленное значение квадратного корня (выводятся на экран).

**Контрольные вопросы**

1. Для чего используются операторы цикла?
2. Из чего состоит цикл?
3. Что называется итерацией?
4. Что называется счетчиком цикла?
5. Какие операторы служат для завершения как текущей итерации, так и цикла в целом?
6. Какие операторы цикла вы знаете? Объясните их различия.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №6**

**«Работа с операторами передачи управления»**

**Цель**: освоить работу с операторами передачи управления.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

В С++ есть четыре оператора, изменяющих естесственный порядок выполнения вычислений:

* оператор безусловного перехода goto;
* оператор выхода из цикла break;
* оператор перехода к следующей итерации цикла continue;
* оператор возврата из функции return.

**Оператор goto**

Оператор безусловного перехода goto имеет формат:

goto метка;

В теле той же функции должна присутствовать ровно одна конструкция вида:

метка: оператор;

Оператор goto передает управление на помеченный оператор. Метка – это обычный идентификатор, областью видимости которого является функция, в теле которой он задан.

**Оператор break**

Оператор break используется внутри операторов цикла или switch для обеспечения перехода в точку программы, находящуюся непосредственно за оператором, внутри которого находится break.

**Оператор continue**

Оператор перехода к следующей итерации цикла continue пропускает все операторы, оставшиеся до конца тела цикла, и передает управление на начало следующей итерации.

**Оператор return**

Оператор возврата из функции return завершает выполнение функции и передает управление в точку ее вызова. Вид оператора:

return [выражение];

Выражение должно иметь скалярный тип. Если тип возвращаемого функцией значения описан как void, выражение должно отсутствовать.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

Напишите программу для вычисления значения гиперболического синуса вещественного аргумента *x* с заданной точностью *Eps* с помощью разложения в бесконечный ряд:

,

Входными данными является число *x* и величина точности *Eps* (вводятся с клавиатуры), выходные данные — вычисленное значение гиперболического синуса (выводятся на экран).

**Контрольные вопросы**

1. Какие вы знаете операторы передачи управления?
2. Напишите формат оператора безусловного перехода goto.
3. Что называется меткой?
4. Для чего служит оператор break?
5. Для чего служит оператор continue?
6. Для чего служит оператор return?

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №7**

**«Создание и работа со статическими одномерными массивами»**

**Цель**: освоить работу со статическими одномерными массивами.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

При использовании простых переменных каждой области памяти для хранения данных соответствует свое имя. Если с группой величин одинакового типа требуется выполнить однообразные действия, им дают одно одно имя, а различают по порядковому номеру. Это позволяет компактно записывать множество операций с помощью циклов. Конечная именованная последовательность однотипных величин называется массовом. Описание массива в программе отличается от описания простой переменной наличием после имени квадрантных скобок, в которых задается количество элементов массива (размерность):

float a[10];

Элементы массива нумеруются с нуля. При описании массива используются те же модификаторы, что и для простых переменных. Инициализирующие значения для массивов записываются в фигурных скобках. Значение элементам присваиваются по порядку.

Для доступа к элементам массива после его имени указывается номер элемента (индекс) в квадрантных скобках.

Размерность массива задается с помощью именованных констант.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Написать программу, подсчитывающую сумму элементов одномерного массива размерностью 10. Элементы массива ввести с клавиатуры. Результат вывести на экран.

**Вариант II**

Написать программу, подсчитывающую произведение элементов одномерного массива размерностью 15. Элементы массива ввести с клавиатуры. Результат вывести на экран.

**Контрольные вопросы**

1. Что называется массивом?
2. Каково назначение массивов?
3. Что называется размерностью массива?
4. Как нумеруются элементы массива?
5. Как производится доступ к элементам массива?

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №8**

**«Создание и работа со статическими двумерными массивами»**

**Цель**: освоить работу со статическими двумерными массивами.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Многомерные массивы задаются указанием каждого измерения в квадрантных скобках, например, оператор

int matr [6][8];

задает описание двумерного массива из 6 строк и 8 столбцов. В памяти такой массив располагается в последовательных ячейках построчно. Многомерные массивы размещаются так, что при переходе к следующему элементу быстрее всего изменяется последний индекс. Для доступа к элементу многомерного массива указываются все его индексы, например, matr [i][j].

При инициализации многомерного массива он представляется либо как массив из массивов, при этом каждый массив заключается в свои фигурные скобки (в этом случае левую размерность при описании можно не указывать), либо задается общий список элементов в том порядке, в котором элементы располагаются в памяти.

Для создания динамического многомерного массива необходимо указать в операции new все его размерности (самая левая размерность может быть переменной), например:

int n = 5;

int \*\* m = (int \*\*) new int [n][10];

Более безопасный и универсальный способ выделения памяти под двумерный массив, когда обе его размерности задаются на этапе выполнения программы, приведен ниже:

int a, b;

cout << «Введите количество строк и столбцов»;

cin >> a >> b;

int \*\*a = new int \*[a];

for (int i = 0; i < a; i++)

{

a[i] = new int [b];

}

Освобождение памяти из-под массива с любым количеством измерений выполняется с помощью операции delete []. Указатель на константу удалять нельзя.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Найти сумму всех диагональных элементов двумерного массива размерностью 6x6. Элементы массива ввести с клавиатуры, результат вычислений вывести на экран.

**Вариант II**

Найти произведение всех диагональных элементов двумерного массива размерностью 6x6. Элементы массива ввести с клавиатуры, результат вычислений вывести на экран.

**Контрольные вопросы**

1. Что называется двумерным массивом?
2. Каково назначение двумерных массивов?
3. Как производится доступ к элементам двумерного массива?
4. Как создаются динамические двумерные массивы?

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №9**

**«Сортировка одномерных массивов»**

**Цель**: освоить работу со статическими двумерными массивами.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Сортировка целочисленного массива методом выбора.

Алгоритм сортировки состоит в том, сто выбирается наименьший элемент массива и меняется местами с первым элементом, затем рассматриваются элементы, начиная со второго, и наименьший из них меняется местами со вторым элементом, итак далее n-1 раз (при последнем проходе цикла при необходимости меняются местами предпоследний и последний элементы массива). Процесс обмена происходит через буферную переменную.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получит задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Написать программу для сортировки методом выбора одномерного массива размерностью 10. Результат сортировки вывести на экран, элементы массива ввести с клавиатуры.

**Вариант II**

Написать программу для сортировки методом выбора одномерного массива размерностью 10. Результат сортировки вывести на экран, элементы массива ввести с клавиатуры.

**Контрольные вопросы**

1. Для чего применяется сортировка массивов?
2. В чем заключается принцип сортировки массива методом выбора?
3. Нарисуйте блок-схему сортировки массива методом выбора.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №10**

**«Поиск минимального значения функции на заданном промежутке»**

**Цель**: освоить работу со динамическими двумерными массивами.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

При инициализации многомерного массива он представляется либо как массив из массивов, при этом каждый массив заключается в свои фигурные скобки (в этом случае левую размерность при описании можно не указывать), либо задается общий список элементов в том порядке, в котором элементы располагаются в памяти.

Для создания динамического многомерного массива необходимо указать в операции new все его размерности (самая левая размерность может быть переменной), например:

int n = 5;

int \*\* m = (int \*\*) new int [n][10];

Более безопасный и универсальный способ выделения памяти под двумерный массив, когда обе его размерности задаются на этапе выполнения программы, приведен ниже:

int a, b;

cout << «Введите количество строк и столбцов»;

cin >> a >> b;

int \*\*a = new int \*[a];

for (int i = 0; i < a; i++)

{

a[i] = new int [b];

}

Освобождение памяти из-под массива с любым количеством измерений выполняется с помощью операции delete []. Указатель на константу удалять нельзя.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Написать программу для нахождения наименьшего значение функции на интервале [-9; 15], разбив его на *n* частей. Число *n* вводится с клавиатуры. Результат вывести на экран.

**Вариант II**

Написать программу для нахождения наименьшего значение функции на интервале [-19; 20], разбив его на *n* частей. Число *n* вводится с клавиатуры. Результат вывести на экран.

**Контрольные вопросы**

1. Для чего применяется динамические массивы?
2. Опишите алгоритм поиска наименьшего значения?
3. Нарисуйте блок-схему поиска наименьшего значения.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №11**

**«Работа с функциями»**

**Цель**: освоить работу с функциями.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Функция – это именованная последовательность описаний и операторов, выполняющая какое-либо законченное действие. Функция может принимать параметры и возвращать значение.

Любая программа на С++ состоит из функций, одна из которых должна иметь имя main (с нее начинается выполнение любой программы). Функция начинает выполняться в момент вызова. Любая функция должна быть объявлена и определена.

Объявление функции задает ее имя, тип возвращаемого значения и список передаваемых параметров. Опеределение функции содержит, кроме объявления, тело функции, представляющее собой последовательность операторов и описаний в фигурных скобках:

[класс] тип имя ([ список\_параметров])[throw (исключения) ]

{тело функции}

Составные части определения.

* С помощью необязательного модификатора *класс* можно явно задать область видимости функции, используя ключевые слова extern и static.
* Тип возвращаемого функцией значения может быть любым, кроме массива и функции. Если функция не должна возвращать значение, указывается тип void.

В определении, в объявлении и при вызове одной и той же функции типы и порядок следования параметров должны совпадать.

Для вызова функции в простейшем случае нужно указать ее имя, за которым в круглых скобках через запятую перечисляются имена передаваемых аргументов. Вызов функции может находиться в любом месте программы, где по синтаксису допустимо выражение того типа, который формирует функция.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Написать функция, вычисляющую значение выражения . Произвести ее вызов из функции main() и вывести на экран значение y(1), y(2), y(10).

**Вариант II**

Написать функция, вычисляющую значение выражения . Произвести ее вызов из функции main() и вывести на экран значение y(10), y(-2), y(1).

**Контрольные вопросы**

1. Что называется функцией?
2. Из чего состоит программа на языке С++?
3. Что называется объявлением функции?
4. Напишите структуру функции.
5. Какие типы возвращаемого значения могут быть у функции?

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №12**

**«Передача массивов в функцию в качестве параметров»**

**Цель**: научиться передавать массивы в функцию в качестве параметров.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

При использовании в качестве параметра массива в функцию передается указатель на его первый элемент, иными словами, массив всегда передается по адресу. При этом информация о количестве элементов массива теряется, и следует передавать его размерность через отдельный параметр (в случае массива символов, то есть строки, ее фактическую длину можно определить по положению нуль-символа).

При передаче многомерных массивов все размерности, если они не известны на этапе компиляции, должны передаваться в качестве парметров. Внутри функции массив интерпретируется как одномерный, а его индекс пересчитывается в программе.

Для того чтобы работать с двумерным массивом естественным образом, можно применить альтернативный способ выделения памяти: сначала память выделяется под столбец указателей на строки массива, а затем в цикле под каждую строку. Освобождение памяти должно выполняться в обратном порядке.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Написать функцию, которая возвращает сумму элементов целочисленного одномерного массива в которую заданный массив передается в качестве параметра. Произвести вызов ее из функции main() и вывести на экран найденную сумму. Элементы массива вводятся с клавиатуры, размерность массива 10.

**Вариант II**

Написать функцию, которая возвращает сумму элементов целочисленного одномерного массива в которую заданный массив передается в качестве параметра. Произвести вызов ее из функции main() и вывести на экран найденную сумму. Элементы массива вводятся с клавиатуры, размерность массива 15.

**Контрольные вопросы**

1. Как производится передача одномерных массивов в качестве параметров?
2. Как производится передача многомерных массивов в качестве параметров?
3. Назовите способ выделения памяти для работы с двумерным массивом в функции.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

## Практическая работа №13

**«Работа с рекурсивными функциями»**

**Цель**: освоить работу с рекурсивными функциями.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Рекурсивной называется функция, которая вызывает саму себя. Такая рекурсия называется прямой. Существует еще косвенная рекурсия, когда две или более функций вызывают друг друга. Если функция вызывает саму себя, в стеке создается копия значений ее параметров, как и при вызове обычной функции, после чего управление передается первому исполняемому оператору функции. При повторном вызове этот процесс повторяется. Для завершения вычислений каждая рекурсивная функция должна содержать хотя бы одну нерекурсивную ветвь алгоритма, заканчивающуюся оператором возврата. При завершении функции соответствующая часть стека освобождается, и управление передается вызывающей функции, выполнение которой продолжается с точки, следующей за рекурсивным вызовом.

Рекурсивные функции чаще всего применяются для компактной реализации алгоритмов, а также для работы со структурами данных, описанными рекурсивно, например, двоичными деревьями. Любую рекурсивную функцию можно реализовать без применения рекурсии, для этого программист должен обеспечить хранение всех необходимых данных самостоятельно. Достоинством рекурсии является компактная запись, а недостатками — расход времени и памяти на повторные вызовы функции и передачу ей копий всех параметров, и, главное, опасность переполнения стека.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

С использованием рекурсивных функций написать программу, вычисляющую факториал числа (следует помнить, что 0! = 1 и 1! = 1), которое вводится с клавиатуры. Результат вычислений вывести на экран.

**Контрольные вопросы**

1. Какая функция называется рекурсивной?
2. Какие существуют виды рекурсий, в чем их различие?
3. Каков алгоритм работы рекурсивной функции?
4. Где применяются рекурсивные функции?
5. Какими достоинствами и недостатками обладают рекурсивные функции?

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

## Практическая работа №14

**«Работа с файлами»**

**Цель**: освоить работу с файлами.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows XP, IDE Borland C++.

**Краткие сведения из теории**

Физический поток представляет собой файл или устройство.

Работа с потоком начинается с его открытия. Поток можно открывать для чтения и/или записи в двоичном и текстовом режиме. Функция открытия потока имеет формат:

FILE\* fopen (const char\* filename, const char\* mode);

При успешном открытии потока функция возарвщает указатель на предопределенную структуру типа FILE, содержащую всю необходимую для работы с потоком информацию, или NULL в противном случае. Первый параметр — имя открытого файла в виде С-строки, второй — режим открытия файла:

* «r» – файл открыт для чтения;
* «w» – открывается пустой файл для записи (если файл существует, он стирается);
* «а» – файл открывается для добавления информации в его конец;
* «r+» – файл открывается для чтения и записи (файл должен существовать);
* «w+» – открывается пустой файл для чтения и записи (если файл существует, он стирается;
* «а+» – файл открывается для чтения и добавления информации в его конец.

Режим открытия также может содержать символы t (текстовый режим) или b (двоичный режим), отличающиеся обработкой символов перехода на новую строку. По умолчанию файл открывается в текстовом режиме.

Ввод/вывод в поток можно осуществить различными способами: в виде последовательных байтов, в виде символов и строк или с использованием форматных преобразователей.

Операции ввода/выводы выполняются с текущей позиции потока, определяемой положением указателя потока. Указатель устанавливается при открытии на начало или конец файла (в соответствии с режимом открытия) и изменяется автоматически после каждой операции ввода/вывода. Текущее положение указателя можно получить с помощью функций ftell и fgetpos и задать явным образом с помощью функций fseek и fsetpos. Эти функции нельзя использовать для стандартных потоков. Ниже перечислены основные функции ввода/вывода потока.

* Чтение и запись потока байтов выполняют функции fread и fwrite.
* Чтение символа из потока — getc, fgetc, из стандартного потока stdin — getchar.
* Запись символа в поток — putc, fputc, в стандартный поток stdout — putchаr.
* Чтение строки из потока — fgets, из стандартного потока stdin — gets.
* Запись строки в поток — fputs, в стандартный поток stdout — puts.
* Форматированный ввод из потока — fscanf, из стандартного потока stdin — scanf, из строки — sscanf.
* Форматированный вывод в поток — fprintf, в стандартный поток stdout — printf, в строку — sprintf.

Поток закрывается либо при завершении работы программы, либо явным образом с помощью функции fclose:

int fclose (FILE\*);

Перед закрытием потока информация из связанных с ним буферов выгружается на диск. Рекомендуется всегда явным образом закрывать потоки, открытые для записи, чтобы избежать потери данных.

Функции работы с потоком возвращает значения, которые рекомендуется анализировать в программе и обрабатывать ошибочные ситуации, возникающие, например, при открытии существующих файлов или чтении из потока. При работе с файлами часто используют функции feof и ferror:

int feof(FILE\*) -- возвращает не равное нулю значение, если достигнут конец файла, в противном случае 0;

int ferror (FILE\*) -- возвращает не равное нулю значение, если обнаружена ошибка ввода/вывода, в противном случае 0.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Написать программу, которая создает новый текстовый файл и записывает в него следующие данные (данные вводятся с клавиатуры):

open

file

1

2

3

close

file

**Вариант II**

Написать программу, которая создает новый текстовый файл и записывает в него следующие данные (данные вводятся с клавиатуры):

open1

file1

11

21

31

close1

file1

**Контрольные вопросы**

1. Что называет потоком?
2. Назовите синтаксис функции открытия потока.
3. Какие существуют режимы открытия файла?
4. Напишите пример функции, которая открывает файл только для чтения.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

**Практическая работа №15**

**«Работа со строками и символами»**

**Цель**: освоить работу со строками и символами.

**Материально-техническое оснащение**: персональный компьютер.

**Программное обеспечение**: ОС Windows, IDE Borland C++.

**Краткие сведения из теории**

Строка представляет собой массив символов, заканчивающийся нуль-символом. В С++ есть две возможности работы со строками: функции, унаследованные из библиотеки С (заголовочный файл <string.h> или <cstring>), и библиотечный класс С++ string, предоставляющий более широкие возможности представления, обработки и контроля строк.

Библиотека С содержит функции копирования строк (strcpy, strncpy), сравнение (strcmp и strncmp), объединение строк (strcat, strncat), поиска подстроки (strstr), поиска вхождения символов (strchr, strrchr, strpbrk), определение длины строки (strlen) и другие.

В заголовочных файлах <stdlib.h> и <cstdlib> содержатся полезные функции преобразования строк в числа (обратные преобразования можно сделать с помощью функции sprintf):

double alof (const char \*p) — преобразует переданную строку в double.

int atoi (const char \*p) — преобразует переданную строку в int.

long atol (const char \*p) — преобразует переданную строку в long.

Пробелы и табуляция в начале строки пропускаются. Преобразование прекращается при встрече недопустимого символа или конца строки. Если строку нельзя преобразовать в число, возвращается ноль. Если число выходит за пределы диапазона данного типа, переменной errno (заголовочный файл <cerrno>) присваивается значение ERANGE и возвращается допустимое число.

**Порядок выполнения практической работы**

1. Повторить теоретический материал по данной теме.
2. Получить допуск к практической работе.
3. Получить задание.
4. Выполнить задание.
5. Подготовить ответы на теоретические вопросы.
6. Оформить отчет.
7. Защитить практическую работу.

**Задание**

**Вариант I**

Дана строка «2, 38, 5, 56, 98, 0, 0, 1, 1, 56, 98, 23». Напишите программу, которая заполняет числами массив типа double из этой строки. Результаты вывести на экран.

**Вариант II**

Дана строка «12, 328, 15, 526, 8, 10, 20, 12, 13, 556, 988, 293». Напишите программу, которая заполняет числами массив типа double из этой строки. Результаты вывести на экран.

**Контрольные вопросы**

1. Что представляет из себя строка?
2. Какие функции содержит библиотек С для работы со строками?
3. Напишите синтаксис функций преобразования строк в числа.
4. Напишите программу, которая преобразует строку в число, если преобразование невозможно, программа должна вывести на экран соответствующее сообщение.

**Содержание отчета**

1. Тема занятия.
2. Цель работы.
3. Предъявление выполненного задания преподавателю.
4. Текст программы.

# ВОПРОСЫ К ЗАЧЕТУ

1. Состав языка С++.
2. Алфавит языка С++.
3. Константы, типы констант.
4. Типы данных С++.
5. Структура программы в С++.
6. Переменные, виды переменных.
7. Область действия переменных, время жизни.
8. Виды операций в языке С++.
9. Базовые конструкции структурного программирования.
10. Основные виды конструкций.
11. Оператор ветвления if: назначение, синтаксис.
12. Оператор ветвления switch: назначение, синтаксис.
13. Оператор цикла с предусловием: назначение, синтаксис.
14. Оператор цикла с постусловием: назначение, синтаксис.
15. Оператор цикла с параметром: назначение, синтаксис.
16. Оператор передачи управления goto: назначение, синтаксис.
17. Указатели: понятие указателя.
18. Назначение и виды указателей.
19. Операции с указателями.
20. Ссылки: определение и назначение.
21. Одномерные массивы: определение, назначение.
22. Описание одномерного массива, нумерация элементов, ввод массива.
23. Работа с элементами одномерного массива.
24. Динамические одномерные массивы: определение, создание, работа с элементами одномерных динамических массивов.
25. Многомерные массивы: определение, создание статического двумерного массива.
26. Работа с элементами многомерного массива.
27. Динамические одномерные массивы: создание и работа с элементами двумерных динамических массивов.
28. Строки: определение, способы работы со строками.
29. Функции: определение, назначение.
30. Базовая структура функций в С++, тип возвращаемого значение,
31. Использование глобальных переменных в функции.
32. Параметры функции в С++, виды параметров.
33. Способы передачи параметров в функцию.
34. Передача массивов в качестве параметров в функцию.
35. Функции с переменным числом параметров.
36. Рекурсивные функции: определение, виды рекурсий.
37. Структура рекурсивной функции.
38. Область применения рекурсивных функций.
39. Функция main(): форматы и параметры функции main().
40. Функции стандартной библиотеки.
41. Директивы препроцессора.
42. Область действия идентификаторов, виды областей действия.
43. Особенности и рекомендации для создания программ.
44. Этапы создания программ.
45. Постановка задачи и разработке программы.
46. Разработка внутренних структур данных.
47. Структурное программирование.
48. Нисходящее тестирование программы.
49. Динамические структуры данных.
50. Виды динамических структур данных.
51. Элемент динамической структуры данных, его состав.
52. Линейные списки: определение, назначение.
53. Основные операции на списком.
54. Стеки: определение, назначение.
55. Основные операции над стеками.
56. Очереди: определение, назначение.
57. Основные операции над очередью.
58. Бинарные деревья: определение, структура.
59. Операции над бинарными деревьями.