Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Шебзухова Татьяна Александровна
Должность: Директор Пятигорского института (филиал) Северо-Кавказского федерального университета
Дата подписания: 13.06.2024 16:10:33
Уникальный программный ключ:
d74ce93cd40e39275c3ba2f58486412a1c8ef96f

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»
Пятигорский институт (филиал) СКФУ
Колледж Пятигорского института (филиал) СКФУ

ПМ.05 ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ИНФОРМАЦИОННЫХ СИСТЕМ
ПМ.05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ЛАБОРАТОРНЫХ РАБОТ

Специальности СПО

09.02.07 Информационные системы и программирование

Пятигорск 2024

Методические указания для лабораторных работ по дисциплине ПМ.05.02 Разработка кода информационных систем составлены в соответствии с требованиями ФГОС СПО. Предназначены для студентов, обучающихся по специальности 09.02.07 Информационные системы и программирование.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Программа дисциплины Разработка кода информационных систем предусматривает изучение алгоритмизации как базовой составляющей технологического процесса создания программного продукта, а так же изучение языка программирования и формирование навыков разработки ИС, объектно-ориентированного программирования у студентов. При изучении предмета следует соблюдать единство терминологии и обозначения в соответствии с действующими стандартами, Международной системной единицы (СИ).

В результате изучения дисциплины студенты *должны:*

знать:

- основные виды и процедуры обработки информации, модели и методы решения задач обработки информации;
- основные платформы для создания, исполнения и управления информационной системой;
- основные процессы управления проектом разработки;
- основные модели построения информационных систем, их структуру, особенности и области применения;
- методы и средства проектирования, разработки и тестирования информационных систем;
- систему стандартизации, сертификации и систему обеспечения качества продукции.

уметь:

- осуществлять постановку задач по обработке информации;
- проводить анализ предметной области;
- осуществлять выбор модели и средства построения информационной системы и программных средств;
- использовать алгоритмы обработки информации для различных приложений;
- решать прикладные вопросы программирования и языка сценариев для создания программ;
- разрабатывать графический интерфейс приложения;
- создавать и управлять проектом по разработке приложения;
- проектировать и разрабатывать систему по заданным требованиям и спецификациям. **иметь практический опыт в**:
- управлении процессом разработки приложений с использованием инструментальных средств;
- обеспечении сбора данных для анализа использования и функционирования информационной системы;
- программировании в соответствии с требованиями технического задания;
- использовании критериев оценки качества и надежности функционирования информационной системы;
- применении методики тестирования разрабатываемых приложений;

-определении состава оборудования и программных средств разработки информационной системы;

- разработке документации по эксплуатации информационной системы;
- проведении оценки качества и экономической эффективности информационной системы в рамках своей компетенции;
- модификации отдельных модулей информационной системы.

В результате освоения учебной дисциплины студент должен овладевать:

Общими компетенциями:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 09. Использовать информационные технологии в профессиональной деятельности.

ОК 10. Пользоваться профессиональной документацией на государственном и иностранном языке.

Профессиональными компетенциями:

ПК 5.1. Собирать исходные данные для разработки проектной документации на информационную систему.

ПК 5.2. Разрабатывать проектную документацию на разработку информационной системы в соответствии с требованиями заказчика.

ПК 5.3. Разрабатывать подсистемы безопасности информационной системы в соответствии с техническим заданием.

ПК 5.4. Производить разработку модулей информационной системы в соответствии с техническим заданием.

ПК 5.5. Осуществлять тестирование информационной системы на этапе опытной эксплуатации с фиксацией выявленных ошибок кодирования в разрабатываемых модулях информационной системы.

ПК 5.6. Разрабатывать техническую документацию на эксплуатацию информационной системы.

ПК 5.7. Производить оценку информационной системы для выявления возможности ее модернизации.

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

По дисциплине Разработка кода информационных систем лабораторные работы содержат задачи и теоретические вопросы. Варианты для каждого обучающегося - индивидуальные.

Задачи и ответы на вопросы, выполненные не по своему варианту, не засчитываются.

Лабораторная работа выполняется в отдельной тетради. Условия задачи и формулировки вопросов переписываются полностью. Формулы, расчеты, ответы на вопросы пишутся ручкой, а чертежи, схемы и рисунки выполняются карандашом, на графиках и диаграммах указывается масштаб. Вначале задача решается в общем виде, затем делаются расчёты по условию задания. Решение задач обязательно ведется в Международной системе единиц (СИ).

При выполнении лабораторной работы необходимо следовать методическим указаниям: повторить краткое содержание теории, запомнить основные формулы и законы, проанализировать пример выполнения аналогичного задания, затем преступить непосредственно к решению задачи. К зачету допускаются студенты, получившие положительные оценки по всем лабораторным работам.

Правила выполнения лабораторной работы.

1. Студент должен прийти на лабораторную работу подготовленным к выполнению лабораторной работы.
2. Каждый студент после проведения работы должен представить отчет о проделанной работе с анализом полученных результатов и выводом по работе.
3. Таблицы и рисунки следует выполнять с помощью чертежных инструментов (линейки, циркуля, и.т.д.) карандашом с соблюдением ЕСКД.
4. Расчет следует проводить с точностью до двух значащих цифр.
5. Исправления проводить на обратной стороне листа. При мелких исправлениях неправильное слово (буква, число и т.п.) аккуратно зачеркивается и над ним пишут правильное пропущенное слово (букву, число и т.п.).
6. Вспомогательные расчеты можно выполнять на отдельных листах, а при необходимости на листах отчета.
7. Если студент не выполнит лабораторную работу или часть работы, то он выполнит ее во внеурочное время, согласованное с преподавателем.
8. Оценку по лабораторной работе студент получает с учетом срока выполнения работы, если;

- расчеты выполнены правильно и в полном объеме;
- сделан анализ проделанной работы и вывод по результатам работы;
- студент может пояснить выполнение любого этапа работы;
- отчет выполнен в соответствии с требованиями к выполнению работы.

Лабораторная работа №1. Ввод текста программы. Сохранение программы

Цель: Получение базовых навыков использования системы программирования, необходимых для дальнейшего выполнения цикла лабораторных работ.

Основные понятия: Интегрированная среда, компиляция, отладчик.

Теоретическая часть.

1. Запуск системы программирования

Система программирования MS Visual studio состоит из большого числа файлов, хранение которых структурировано по каталогам.

Для запуска системы программирования достаточно щелкнуть на ярлыке на рабочем столе.

2. Пользование подсказкой в среде

Система программирования MS Visual studio имеет мощную систему онлайновой подсказки, которая очень полезна, особенно для начинающего программиста.

Подсказка является гипертекстовой, то есть, выбрав одно из выделенных слов, можно получить подсказку, связанную с этим словом.

Первая возможность обращения к подсказке - через **Главное Меню -> Help**. В том меню самыми полезными являются пункты Contents и Index.

Пункт **Content**s выводит на экран тематический список разделов, по которым можно получить подсказку. Пункт **Inde**x выводит на экран алфавитный список всех ключевых слов и имен функций языка и системы программирования MS Visual studio. Выбирая пункт этого списка, можно получить подробную подсказку по нему.

Другая возможность обращения к подсказке - через клавиши **F1** и комбинацию клавиш **Ctrl+F1**. Эта подсказка - контекстно-зависимая, то есть, подсказка, которая появляется на экране, относится именно к тому состоянию системы программирования, в котором она сейчас находится.

По клавише **F1** получаете подсказку по активному на данный момент меню или окну. Комбинацией **Ctrl+F1** можно пользоваться только в активном окне редактора. Если при нажатии этой комбинации курсор находится на каком-то ключевом слове или на функции, то на экран выводится подробная подсказка по этому ключевому слову или функции.

3. Создание, редактирование и сохранение программы

Выберите **Главное Меню -> File -> New**. У вас на экране откроется пустое окно редактора с заголовком NONAME.CPP. В этом окне наберите текст программы.

Текст, который Вы набираете нужно сохранить текст в файле на внешней памяти. Для этого выполните: **Главное Меню -> File -> Save as...** . При этом у Вас на экране появляется окно Save File As. В верхнем поле этого окна наберите имя, под которым Вы хотите сохранить текст программы без расширения. Система автоматично добавит к имени Вашей программы расширение CPP - стандартное расширение для программ, написанных на языке C++.

Далее после каждого добавления нескольких строк или при внесении некоторого количества изменений в текст сохраняйте программу под тем же именем. Для этого достаточно выполнить: **Главное Меню -> File -> Save** или нажать клавишу **F2**.

Впоследствии, если Вам нужно будет снова загрузить в редактор текст той же программы, выполните: **Главное Меню -> File -> Open** (или нажмите клавишу **F3**). Вы получите окно Open a File, в котором надо выбрать нужный файл. Файл, который Вы выбрали, откроется в окне редактора.

4. Компиляция и выполнение программы

Закончив набор текста и сохранив файл программы на диске, выполните компиляцию программы. Для этого выполните: **Главное Меню -> Compile -> Compile** или нажмите комбинацию клавиш **Alt+F9**. На экране появляется окно Compiling, в котором отображается ход

компиляции. При нормальном завершении компиляции в нижней строке этого окна должно быть выведено:

Success: Press any key

Если там выводится:

Errors: Press any keyили:

Warnings: Press any key

- ваша программа требует коррекции.

После того, как Вы откомпилировали программу без ошибок, запустите ее на выполнение. Для этого выполните: **Главное Меню -> Run -> Run** или нажмите комбинацию клавиш **Ctrl+F9**. На экране появляется окно Linking, в котором отображается ход компоновки. При нормальном завершении компоновки это окно пропадает само собой и выполняется программа. Если же в этому окне выводится:

Errors: Press any key- ваша программа требует коррекции.

5. *Отладка программы*

В системе программирования есть отладчик, который работает на уровне текстового кода. Используя его, Вы можете выполнять программу в пошаговом режиме, устанавливать точки останова выполнения и следить за текущими значениями переменных программы.

Выполните: **Главное Меню -> Debug -> Watches -> Add watch** (или нажмите комбинацию клавиш **Ctrl+F7**). В окне Add Watch, которое появится на экране, введите в поле Watch Expression имена переменных, текущие значения которых надо отслеживать.

Клавишей **F8** запустите программу в пошаговом режиме. Каждое следующее нажатие клавиши F8 будет продвигать выполнение программы на оператор вперед с изменением значения переменных.

Задание

Изучить основные приемы работы в среде MS Visual studio.

Осуществить основные этапы разработки программ.

На оценку «Удовлетворительно»

Разработать программу-калькулятор для выполнения четырех основных арифметических действия (сложение, вычитание, умножение, деление) над двумя вводимых с клавиатуры величинами.

На оценку «Хорошо»

Разработать программу, которая по вводимому с клавиатуры радиусу вычисляет окружность и площадь получаемого круга. *На оценку «Отлично»*

Разработать программу, которая по вводимым с клавиатуры радиусу основания и высоты цилиндра вычисляет его объем.

Контрольные вопросы

1. Перечислите этапы разработки задачи.
2. Как можно получить справку по нужной команде языка?
3. Что такое компиляция?
4. Приведите примеры ошибок компиляции. 5. Перечислите, какие файлы образуется после каждого этапа выполнения программы.

Лабораторная работа №2. Программа математических расчетов

Цель: Получение навыков в работе с арифметическими выражениями и стандартными математическими функциями C++.

Темы для предварительной проработки Стандартные функции C++.

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Структура программы на языке С/С++. Основные этапы разработки программ.

Теоретический материал

В С++ определен широкий набор встроенных операторов, которые дают в руки программисту мощные рычаги управления при создании и вычислении разнообразнейших выражений. Оператор (operator) — это символ, который указывает компилятору на выполнение конкретных математических действий или логических манипуляций. В С++ имеется четыре общих класса операторов: арифметические, поразрядные, логические и операторы отношений. Помимо них определены другие операторы специального назначения.

1. Арифметические операторы

В таблице 1 перечислены арифметические операторы, разрешенные для применения в С++. Действие операторов +, -, * и / совпадает с действием аналогичных операторов в любом другом языке программирования. Их можно применять к данным любого встроенного числового типа. После применения оператора деления (/) к целому числу остаток будет отброшен.

Таблица 1. Арифметические операторы

Оператор	**Действие**
+	Сложение
-	Вычитание, а также унарный минус
*	Умножение
/	Деление
%	Деление по модулю
--	Декремент
++	Инкремент

Остаток от деления можно получить с помощью оператора деления по модулю (%). В С++ оператор "%" нельзя применять к типам с плавающей точкой (float или double.

Унарный минус представляет собой умножение значения своего единственного операнда на -1. Операция ++ увеличивает единственный операнд на 1, а – наоборот уменьшает на 1.

2. Математические функции (заголовочный файл math. h)

В таблице 2. собраны основные математические функции. Все они имеют вещественные аргументы и возвращаемый результат.

Таблица 2. Математические функции

Обращение	**Функция**
acos (x)	арккосинус (радианы)
asin (x)	арксинус (радианы)
atan (x)	арктангенс (радианы)
cos (x)	косинус (x в радианах)
exp (x)	e^x — экспонента от x

fabs (x)	абсолютное значение вещественного x
log (x)	логарифм натуральный — Ln x
log10 (x)	логарифм десятичный — Lg x
pow (x,y)	x в степени y — x^y
sin (x)	синус (x в радианах)
sinh (x)	гиперболический синус
sqrt (x)	корень квадратный (положительное значение)
tan (x)	тангенс (x в радианах)

Задание

Составить программы, вычисляющие

На

$$\frac{b+\sqrt{b^2+4ac}}{2a}-a^3c+b^{-2}$$

$$\frac{CosX}{\pi-2X}+16X\cdot Cos(XY)-2$$

оценку «Удовлетворительно»

1.

2.

На оценку «Хорошо»

Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b.

На оценку «Отлично»

Полторы кошки съедают за полтора часа полторы мышки. Сколько мышек съедят X кошек за Y часов.

Контрольные вопросы

1. Какую необходимо подключить библиотеку для использования в программе стандартных математических функций?
2. Чем отличается результат операции деление для целых и вещественных чисел? 3. Что является результатом операции %? Приведите примеры.

Лабораторная работа №3. Программа исследования базовых типов.

Цель: Исследовать систему стандартных типов языка программирования C/C++.

Темы для предварительной проработки

Арифметические операторы C++ Стандартные функции C++.

Объявление и инициализация переменных.

Структура программы на языке C/C++.

Базовые типы языка C/C++.

Теоретический материал

1. Ситема типов в языке C/C++

В Си++ имеется четыре базовых арифметических (числовых) типа данных. Из них два целочисленных — char, int — и два плавающих (вещественных) — float и double. Кроме того, в программах можно использовать некоторые модификации этих типов, описываемых с помощью служебных слов — модификаторов. Существуют два модификатора размера — short (короткий) и long (длинный) — и два модификатора знаков — signed (знаковый) и unsigned (беззнаковый). Знаковые модификаторы применяются только к целым типам.

Как известно, тип величины связан с ее формой внутреннего представления, множеством принимаемых значений и множеством операций, применимых к этой величине. В таблице перечислены арифметические типы данных Си++, указан объем занимаемой памяти и диапазон допустимых значений.

Размер типа int и unsigned int зависит от размера слова операционной системы, в которой работает компилятор Си++. В 16-разрядных ОС (MS DOS) этим типам соответствуют 2 байта, в 32разрядных (Windows) — 4 байта.

Таблица 3. Типичные размеры значений и диапазоны представления типов

Таблица 4.1

Тип данных	Размер (байт)	Диапазон значений	Эквивалентные названия типа
char	1	−128...+127	signed char
int	2/4	зависит от системы	signed, signed int
unsigned char	1	0...255	нет
unsigned int	2/4	зависит от системы	unsigned
short int	2	−32768...32767	short, signed short int
unsigned short	2	0...65535	unsigned short int
long int	4	−2147483648...2147483647	long, signed long int
unsigned long int	4	0...4294967295	unsigned long
float	4	±(3.4E−38...3.4E+38)	нет
double	8	±(1.7E−308...1.7E+308)	нет
long double	10	±(3.4E−4932...1.1E+4932)	нет

2. Особенности типов в C/C++

Анализируя данные таблицы, можно сделать следующие выводы:

если не указан модификатор знаков, то по умолчанию подразумевается signed;

с базовым типом float модификаторы не употребляются; модификатор short применим только к базовому типу int.

Особенность - тип char причислен к арифметическим типам. В C/C++ величины типа char могут рассматриваться в программе и как символы, и как целые числа. Все зависит от контекста, т.е. от способа использования этой величины. В случае интерпретации величины типа char как символа ее числовое значение является ASCII-кодом. Следующий пример иллюстрирует сказанное.

```
char a=65; cout<<(char)a;      //На экране появится
символ A cout<<(int)a;         //На экране появится
число 65
```

Еще одной особенностью C++ является отсутствие среди базовых типов логического типа данных. В качестве логических величин в C++ выступают целые числа. Интерпретация их значений в логические величины происходит по правилу: равно нулю — ложь, не равно нулю — истина.

В последние версии C++ добавлен отдельный логический тип с именем bool. Его относят к разновидности целых типов данных.

3. Описание переменных в программах на C/C+

+ имя_типа список_переменных; Примеры описаний: char symbol, cc; unsigned char code; int number,row; unsigned long long_number;

float x,X,cc3; double

e,b4; long double

max_num;

Одновременно с описанием можно задать начальные значения переменных. Такое действие называется инициализацией переменных. Описание с инициализацией производится по следующей схеме:

тип имя_переменной = начальное_ значение

Например:

float pi=3.14159, c=1.23;

unsigned int year=2000;

4. Константы в C/C++

Запись целых констант. Целые десятичные числа, начинающиеся не с нуля, например: 4, 356, —128. Целые восьмеричные числа, запись которых начинается с нуля, например: 016, 077. Целые шестнадцатеричные числа, запись которых начинается с символов 0x, например: 0xlA, 0x253, 0xFFFF.

Тип константы компилятор определяет по следующим правилам: если значение константы лежит в диапазоне типа int, то она получает тип int; в противном случае проверяется, лежит ли константа в диапазоне типа unsigned int, в случае положительного ответа она получает этот тип; если не подходит и он, то пробуется тип long и, наконец, unsigned long. Если значение числа не укладывается в диапазон типа unsigned long, то возникает ошибка компиляции.

Запись вещественных констант. Если в записи числовой константы присутствует десятичная точка (2.5) или экспоненциальное расширение (1E-8), то компилятор рассматривает ее как вещественное число и ставит ей в соответствие тип double. Примеры вещественных констант: 44 . 3.14159 44E0 1.5E-4. Использование суффиксов. Программист может явно задать тип константы, используя для этого суффиксы. Существуют три вида суффиксов: F (f) -float; U(u) —unsigned; L(l) - long (для целых и вещественных констант). Кроме того, допускается совместное использование суффиксов U и L в вариантах ul или LU.

Примеры:

3 .14159F — константа типа float, под которую выделяется 4 байта памяти;

3.14L — константа типа long double, занимает 10 байт;

50000U — константа типа unsigned int, занимает 2 байта памяти;

0LU — константа типа unsigned long, занимает 4 байта;

24242424UL — константа типа unsigned long, занимает 4 байта.

Запись символьных и строковых констант. Символьные константы заключаются в апострофы. Например: 'A', 'a', '5', ' + '. Строковые константы, представляющие собой символьные последовательности, заключаются в двойные кавычки. Например: "rezult", "введите исходные данные".

Особую разновидность символьных констант представляют так называемые управляющие символы. Их назначение — управление выводом на экран. Как известно, такие символы расположены в начальной части кодовой таблицы ASCII (коды от 0 до 31) и не имеют графического представления. В программе на Си они изображаются парой символов, первый из которых ' \ '. Вот некоторые из управляющих символов:

'\n' — переход на новую строку;

'\t' — горизонтальная табуляция; '\

a' — подача звукового сигнала.

Именованные константы (константные переменные). В программе на C/C++ могут использоваться именованные константы. Употребляемое для их определения служебное слово **const** принято называть квалификатором доступа. Квалификатор const указывает на то, что данная величина не может изменяться в течение всего времени работы программы. В частности, она не может располагаться в левой части оператора присваивания. Примеры описания константных переменных: const float pi=3.1415 9;

Определение констант на стадии препроцессорной обработки программы. Еще одной возможностью ввести именованную константу является использование препроцессорной директивы #define в следующем формате:

#define <имя константы> <значение константы>

Например:

#define iMAX 1000

Тип констант явно не указывается и определяется по форме записи. В конце директивы не ставится точка с запятой.

3.5. Определение размера памяти

sizeof - операция вычисления размера (в байтах) для объекта того типа, который имеет операнд. Разрешены два формата операции: **sizeof объект и sizeof (тип).**

Пример программы для исследования типов

Использовать унарную операцию языка C++ sizeof, позволяющую определить размер в байтах области памяти, выделенной для стоящего справа операнда. Описать необходимые переменные.

```
main()
{ float
a=3.14159f;
float b, c, d, e;
b=50000l;          c=78945L;          d=123LU;
cout<<”f”<<a<<”dlina="<<sizeof(a)<<endl;
cout<<"l”<<b<<”dlina="<<sizeof(b)<<endl;
cout<<"L”<<”dlina=”<<sizeof(c)<<endl;
cout<<"LU”<<d<<”dlina=”<<sizeof(d)<<endl;
cout<<"LU”<<d<<”dlina=”<<sizeof(d)<<endl;
e=2.58935E+3;
cout<<"exp”<<e<<”dlina=”<<sizeof(e)<<endl;
e=2.58935E-3;
cout<<"exp”<<e<<”dlina=”<<sizeof(e)<<endl; e=-
2.58935E+3;
cout<<"exp”<<e<<”dlina=”<<sizeof(e)<<endl;
d=24242424UL;
cout<<””24242424”<<d<<”dlina=”<<sizeof(d)<<endl;
d=1234567890UL;
cout<<"1234567890”<<d<<”dlina=”<<sizeof(d)<<endl
;
cout<<"1234567890”<<d<<”dlina=”<<sizeof(d)<<end;
a=3.14L;
cout<<"3.14L”<<a<<”dlina=”<<sizeof(a)<<endl; }
Результат работы f
3.141590 dlina=>4 l
50000.000000 dlina=>4 L
78945.000000 dlina=>4
```

LU 123.000000 dlina=>4
LU 123.00 dlina=>4 exp
2589.350098 dlina=>4 exp
0.002589 dlina=>4 exp -
2589.350098 dlina=>4
24242424 24242424.000000 dlina=>4
1234567890 1234567936.000000 dlina=>4
1234567890 1234567936 dlina=>4
3.14L 3.140000 dlina=>4 **Задание**

На оценку «Удовлетворительно»

Провести исследования базовых типов языка C/C++, объявляя данные различных типов. Смоделироватье ситуации переполнения для целых и плавающих типов данных. Использовать операцию sizeof.

На оценку «Хорошо»

Постоить таблицу типов. Использовать различные типы констант.

На оценку «Отлично»

Изучить особенности har. Осуществить вывод на экран символов и их значение ASCII кода.

Контрольные вопросы 1.

Перечислите базовые типы языка программирования C/C++.

2. С какой целью указывают тип величин в программах? 3.

Укажите, что является результатом операции sizeof.

Лабораторная работа №4. Ввод - вывод данных.

Цель: Получение навыков форматирования данных при вводе и выводе на экран средствами языка C++.

Темы для предварительной проработки Структура

программы на C++.

Типы данных.

Теоретический материал

1. Потоковый ввод-вывод в Си++.

В C++ имеются специфические средства ввода-вывода. Это библиотека классов, подключаемая к программе с помощью файла iostream.h. В этой библиотеке определены в качестве объектов стандартные символьные потоки со следующими именами:

cin — стандартный поток ввода с клавиатуры;

cout — стандартный поток вывода на экран.

Ввод данных интерпретируется как извлечение из потока cin и присваивание значений соответствующим переменным. В C++ определена операция извлечения из стандартного потока, знак которой >>. Например, ввод значений в переменную x реализуется оператором: cin>>x;

Вывод данных интерпретируется как помещение в стандартный поток cout выводимых значений. Выводиться могут тексты, заключенные в двойные кавычки, и значения выражений. Знак операции помещения в поток <<. Примеры использования потокового вывода:

```
cout<<a+b;
cout<<"\nРезультат="<<Y;
cout<<"x="<<x<<" y="<<y<<" z = "<<z<<endl;
```

В выходном потоке можно использовать управляющие символы, как и при использовании функции printf (); перед каждым элементом вывода нужно ставить знак операции <<. Элемент

вывода endl является манипулятором, определяющим перевод курсора на новую строку (действует аналогично управляющему символу \n).

В процессе потокового ввода-вывода происходит преобразование из формы внешнего символьного представления во внутренний формат и обратно. Формат определяются автоматически main()

```
{       float a, b, c, p, s;  cout<<"\
na="; cin>>a; cout<<"\nb=";
cin>>b; cout<<"\nc="; cin>>c;
p=(a+b+c)/2;
s=sqrt(p*(p-a)*(p-b)*(p-c) ) ;
cout<<"\nПлощадь треугольника="<<s;
}
```

2. Использование манипуляторов ввода-вывода

Манипуляторы позволяют встраивать инструкции форматирования в выражение вводавывода. Наиболее часто используемые манипуляторы описаны в таблице 4.

Таблица 4. Основные манипуляторы вывода

Манипулятор	**Назначение**	**Функция**
endl	Выводит символ новой строки и "сбрасывает" поток, т.е. переписывает содержимое буфера, связанного с потоком, на соответствующее устройство	Вывод
flush	"Сбрасывает" поток	Вывод
hex	Устанавливает флаг hex	Ввод-вывод
oct	Устанавливает флаг oct	Ввод-вывод
right	Устанавливает флаг right	Вывод
scientific	Устанавливает флаг scientific	Вывод
setbase (int base)	Устанавливает основание системы счисления равной значению base	Вывод
setfill(int ch)	Устанавливает символ-заполнитель равным значению параметра ch	Вывод
setprecision (int p)	Устанавливает количество цифр точности (после десятичной точки)	Вывод
setw (int w)	Устанавливает ширину поля равной значению параметра w	Вывод

При использовании манипуляторов, которые принимают аргументы, необходимо включить в программу заголовок <iomanip.h>.

Манипулятор используется как часть выражения ввода-вывода. Вот пример программы, в которой показано, как с помощью манипуляторов можно управлять форматированием выводимых данных.

```
main() { cout << setprecision (2) <<
1000.243<<« endl; cout << setw(20) <<
"Всем привет! "; }
```

Результаты выполнения этой программы таковы.

```
1e+003 Всем привет!
```

Задание

На оценку «Удовлетворительно»

Идет к-я секунда суток. Определить, сколько целых часов (H) и целых минут (M) прошло с начала суток. Например, если к= 13257 = 3 х 36000 + 40 х 60 + 57, то H = 3, M= 40. Вывести на экран фразу: «Это...часов...минут». Вместо многоточий поставить вычисленные значения H и M.

На оценку «Хорошо»

Составить программу решения обратной задачи по отношению к предыдущей: дано количество часов и минут, прошедших от начала суток. Определить количество секунд. *На оценку «Отлично»*

Составить программу вычисления объема и площади поверхности куба по данной длине ребра.

Контрольные вопросы

1. Какая команда в языке С++ осуществляет ввод с клавиатуры?
2. Какая команда в языке С++ осуществляет вывод на экран?
3. Перечислите основные манипуляторы вывода. 4. Какую неободимо подключить библиотеку для манипуляторов?

Лабораторная работа №5.
Реализация программ, построенных на базе линейного алгоритма.

Цель: Получение навыков в работе с операторами ввода и вывода информации, с арифметическими операторами С++.

Темы для предварительной проработки Арифметические операторы С++.

Стандартные функции С++.

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Теоретический материал

1. Потоковый ввод-вывод в Си++.

В С++ имеются специфические средства ввода-вывода. Это библиотека классов, подключаемая к программе с помощью файла iostream.h. В этой библиотеке определены в качестве объектов стандартные символьные потоки со следующими именами:

cin — стандартный поток ввода с клавиатуры;

cout — стандартный поток вывода на экран.

Ввод данных интерпретируется как извлечение из потока cin и присваивание значений соответствующим переменным. В С++ определена операция извлечения из стандартного потока, знак которой >>. Например, ввод значений в переменную x реализуется оператором: cin>>x;

Вывод данных интерпретируется как помещение в стандартный поток cout выводимых значений. Выводиться могут тексты, заключенные в двойные кавычки, и значения выражений. Знак операции помещения в поток <<. Примеры использования потокового вывода:

```
cout<<a+b;
cout<<"\nРезультат="<<Y;
cout<<"x="<<x<<" y="<<y<<" z = "<<z<<endl;
```

В выходном потоке можно использовать управляющие символы, как и при использовании функции printf (); перед каждым элементом вывода нужно ставить знак операции <<. Элемент вывода endl является манипулятором, определяющим перевод курсора на новую строку (действует аналогично управляющему символу \n).

В процессе потокового ввода-вывода происходит преобразование из формы внешнего символьного представления во внутренний формат и обратно. Формат определяются автоматически

Задание

На оценку «Удовлетворительно»

Составить программу, вычисляющую площадь боковых поверхностей и объем куба по его стороне, вводимой пользователем.

На оценку «Хорошо»

Составить программу, вычисляющую площадь кольца по вводимым с клавиатуры радиусам.

На оценку «Отлично»

Составить программу, вычисляющую площадь трапеции по вводимым с клавиатуры основания и высоте.

Контрольные вопросы

1. Перечислите правила синтаксиса команды ввода cin.
2. Перечислите правила синтаксиса команды вывода cout. 3.
   Что такое управляющий символ?

Лабораторная работа №6.
Математические расчеты для треугольника.

Цель: Получение навыков вычисления арифметических выражений и применение их к расчетным формулам для треугольника. **Темы для предварительной проработки** Типы данных.

Операторы ввода – вывода данных.

Структура программы на языке C/C++.

Теоретический материал

1. Операции и выражения

Во всех языках программирования под выражением подразумевается конструкция, составленная из констант, переменных, знаков операций, функций, скобок. Выражение определяет порядок вычисления некоторого значения. Если это числовое значение, то такое выражение называют арифметическим.

Опишем набор операций, используемых в C++, а также правила записи и вычисления выражений. Напомним, что операция, применяемая к одному операнду, называется унарной, а операция с двумя операндами — бинарной.

2. Арифметические операции

К арифметическим операциям относятся:

- вычитание или унарный минус;
+ сложение или унарный плюс;
* умножение;
/ деление;
% деление по модулю (аналог Mod в Паскале);
++ унарная операция увеличения на единицу (инкремент);
- - унарная операция уменьшения на единицу (декремент).

Все операции, кроме деления по модулю, применимы к любым числовым типам данных. Операция % применима только к целым числам. Если делимое и делитель — целые числа, то и результат — целое число. Например, значение выражения 5/3 будет равно 2, а при вычислении 1/5 получится 0.

Если хотя бы один из операндов имеет вещественный тип, то и результат будет вещественным. Например, операции 5./3, 5./3., 5/3. дадут вещественный результат 1.6666.

Операции инкремента и декремента могут применяться только к переменным и не могут — к константам и выражениям. Операция ++ увеличивает значение переменной на единицу, операция — уменьшает значение переменной на единицу. Оба знака операции могут записываться как перед операндом (префиксная форма), так и после операнда (постфиксная форма), например: ++x или x. Три следующих оператора дают один и тот же результат:

x=x+1; ++x; x++;

Различие проявляется при использовании префиксной и постфиксной форм в выражениях. Проиллюстрируем это на примерах.

Первый пример: a=3; b=2; d=a++*b++;

Результаты будут такими: a = 4, b= 3, d = 6.

Второй пример: a=3; b=2; d=++a*++b;

Результаты будут такими: a = 4, b = 3, d = 12.

При использовании постфиксной формы операции ++ и -- выполняются после того, как значение переменной было использовано в выражении, а префиксные операции — до использования. Поэтому в первом примере значение переменной d вычислялось как произведение 3 на 2, а во втором — как произведение 4 на 3.

Пример программы математических расчетов

Дано: a, b, c — стороны треугольника. Вычислить S— площадь треугольника, используя формулу Герона: $s = \sqrt{p(p-a)(p-b)(p-c)}$ где p — полупериметр треугольника.

```
#include <math.h>
main()
{ float a, b, c, p, s;
cout<<"\na="; cin>>a;
cout<<"\nb="; cin>>b;
cout<<"\nc="; cin>>c;
p=(a+b+c)/2;
s=sqrt(p*(p-a)*(p-b)*(p-c));
cout<<"\nПлощадь треугольника=<<s;
}
```

Пояснения к программному коду.

В выражении для вычисления площади используется библиотечная функция sqrt() — квадратный корень. Данная функция относится к библиотеке математических функций. Для подключения этой библиотеки к нашей программе используется директива препроцессора #include <math.h>.

В рассматриваемой программе операторы cin и cout реализуют соответственно вывод на экран и ввод исходных данных с клавиатуры. Оператор cout<<"\na=" содержит текст ("a=") и управляющий символ ("\n"). Управляющие символы влияют на расположение на экране выводимых знаков. В результате выполнения этого оператора на экран с новой строки выведутся символы a=.

Оператор cout<<"\nПлощадь треугольника=”<< s; список аргументов состоит из одной переменной s. Ее значение выводится на экран.

В программе имеется оператор cin>>a;

Этот оператор производит ввод числового значения в переменную a. В программе все три величины a, b, c можно ввести одним оператором:

cin>>a>>b>>c;

Если последовательность ввода будет такой: 5 3.2 2.4 <Enter>, то переменные получат следующие значения: a = 5,0, b = 3,2, c = 2,4.

Задание

На оценку «Удовлетворительно»

Вводятся координаты вершин треугольника. Составить программу, вычисляющую длины сторон треугольника, его периметр и площадь.

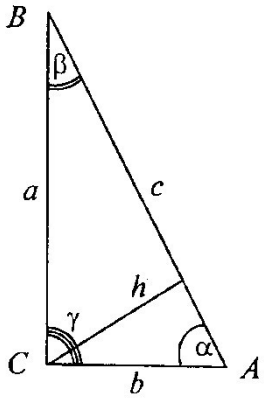
На оценку «Хорошо»

Дополнительно вычислить длины высот, медиан и биссектрис заданного треугольника.

На оценку «Отлично»

Дополнительно вычислить радиусы вписанной и описанной огружностей для треугольника, а так же градусную меру его углов.

Справочный материал



1. Формулы для прямоугольного треугольника

Гипотенуза: $c^2 = a^2 + b^2$

Площадь: $s = \frac{1}{2}ab$

$$\sin A = \frac{a}{c}, \quad \cos A = \frac{b}{c}, \quad tgA = \frac{a}{b}, \quad ctgA = \frac{b}{a}$$

$$\sin^2 \alpha + \cos^2 \alpha = 1, \quad tg\alpha * ctg\alpha = 1$$

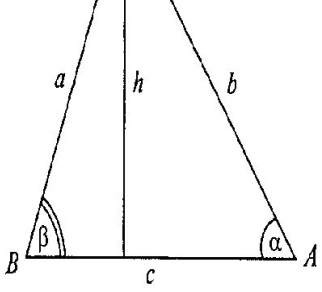
$$tg\alpha = \frac{\sin \alpha}{\cos \alpha}, \quad ctg\alpha = \frac{\cos \alpha}{\sin \alpha}$$

2. Формулы для произвольного треугольника

Площадь произвольного треугольника:

$$s = \frac{1}{2}ab = \frac{1}{2} \sin = \frac{c^2 \sin A \sin B}{2 \sin C} = \frac{h^2 \sin C}{2 \sin A \sin B}$$

Формула Герона для площади треугольника:



$$2$$

$$s = \frac{1}{2} a \sqrt{b^2 - \frac{a^2}{4}}$$

Площадь равнобедренного треугольника:
где a – основание, b – боковая сторона

$$s = \frac{1}{4} a^2 \sqrt{3}$$

Площадь равностороннего треугольника:

Высота к стороне

$$h_a = \frac{2\sqrt{p(p-a)(p-b)(p-c)}}{a}, \qquad = \frac{a+b+c}{2}$$

$$m_a = \frac{1}{2}\sqrt{2b^2 + 2c^2 - a^2}$$

Медиана от вершины A:

$$\beta_a = \frac{2}{b+c}\sqrt{bcp(p-a)}, \qquad = \frac{a+b+c}{2}$$

Биссектриса из угла A:

Теорема косинусов любая сторона a: $a = b^2 + c^2 - 2bc\cos A$, где A – противоположный угол к

,

a:

$$\cos A = \frac{b^2 + c^2 - a^2}{2bc}$$

Теорема синусов:

$$\frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = 2R$$

$$r = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}}, \qquad = \frac{a+b+c}{2}$$

Радиус вписанного в треугольник круга:

стороне a, или

Радиус описанного около треугольника круга:

$$R = \frac{a}{2\sin A} = \frac{abc}{4\sqrt{p(p-a)(p-b)(p-c)}}, \qquad = \frac{a+b+c}{2}$$

Контрольные вопросы

1. Дайте определение динейному алгоритму.
2. Перечислите список команд, обеспечивающих линейный алгоритм.
3. Укажите правила синтаксиса команды ввода и вывода.

Лабораторная работа №7.
Программа выделения цифр из целого числа.

Цель: Получение навыков в работе с арифметическими выражениями и стандартными математическими функциями C++.

Темы для предварительной проработки Арифметические операторы C++.

Стандартные функции C++.

Функции стандартного ввода и вывода.

Теоретический материал

В C++ определен широкий набор встроенных операторов, которые дают в руки программисту мощные рычаги управления при создании и вычислении разнообразнейших выражений. Оператор (operator) — это символ, который указывает компилятору на выполнение конкретных математических действий или логических манипуляций.

Действие операторов +, -, * и / совпадает с действием аналогичных операторов в любом другом языке программирования. Их можно применять к данным любого встроенного числового типа. После применения оператора деления (/) к целому числу остаток будет отброшен.

Остаток от деления можно получить с помощью оператора деления по модулю (%) Этот оператор возвращает остаток от деления нацело. Например, 10 % 3 равно 1. Это означает, что в C++ оператор "%" нельзя применять к типам с плавающей точкой (float или double) Деление по модулю применимо только к целочисленным типам. Использование этого оператора демонстрируется в следующей программе.

Пример программы выделения цифр из числа

```
#include          <stdio.h>
main()
{         long int x=1234567890; cout<<"\n
x => 1234567890";
cout<<"\n (x divide  mod 10000000000)/1000000000 =>"<<
(x%10000000000)/1000000000);
cout<<"\n (x divide  mod 1000000000)/100000000 =>"<<
(x%1000000000)/100000000);
printf("\n (x divide  mod 100000000)/10000000 => "<<
(x%100000000)/10000000); cout<<"\n (x divide  mod 10000000)/1000000 =>
"<<(x%10000000)/1000000); cout<<"\n (x divide  mod 1000000)/100000
=>"<<(x%1000000)/100000); cout<<"\n (x divide mod 100000)/10000
=>"<<(x%100000)/10000); cout<<"\n (x divide  mod 10000)/1000 => "<<(x
%10000)/1000); cout<<"\n (x divide  mod 1000)/100 => "<<(x%1000)/100);
cout<<"\n (x divide  mod 100)/10 => "<<(x%100)/10);
cout<<"\n x divide  mod 10 =>"<<(x%10); cout<<"\
n"; cout<<"\n 2 number";
cout<<"\n (x divide  mod 100000)/1000 => "<<(x%100000)/1000);
cout<<"\n"; cout<<"\n";
cout<<"\n (x divide  mod 10000000)/10000 =>"<<(x%10000000)/10000);
cout<<"\n"; cout<<"\n 4 number";
cout<<"\n (x divide mod 1000000000)/100000=> "<<
(x%1000000000)/100000);
}
```

Результат работы программы: x => 1234567890
(x divide mod 10000000000)/1000000000 => 1
(x divide mod 1000000000)/100000000 => 2
(x divide mod 100000000)/10000000 => 3
(x divide mod 10000000)/1000000 => 4
(x divide mod 1000000)/100000 => 5
(x divide mod 100000)/10000 => 6
(x divide mod 10000)/1000 => 7
(x divide mod 1000)/100 => 8
(x divide mod 100)/10 => 9 x
divide mod 10 => 0
2 number

(x divide mod 100000)/1000 => 67

3 number

(x divide mod 10000000)/10000 => 456

4 number

(x divide mod 1000000000)/100000 => 2345

Задание

На оценку «Удовлетворительно»

Составить программу, которая запрашивает у пользователя четырехзначное число и выдает на экран сумму цифр, из которых состоит исходное число.

На оценку «Хорошо»

Составить программу, которая запрашивает у пользователя трехзначное число и выдает на экран число, полученное перестановкой в обратном порядке цифр, из состоит исходное число. *На оценку «Отлично»*

Составить программу, которая запрашивает у пользователя два двухзначных числа и выдает на экран число, полученное слиянием первого и второго чисел.

Контрольные вопросы

1. Что является результатом операции %?
2. Как получить последнюю цифру, входящую в число?
3. Как получить первую цифру в пятизначном числе?

Лабораторная работа №8.
Вычисление логических выражений.

Цель: Получение навыков вычисления логических выражений.

Темы для предварительной проработки Типы данных.

Операторы ввода – вывода данных.

Приоритет операций.

Структура программы на языке C/C++.

Теоретический материал

1. Операции и выражения

Во всех языках программирования под выражением подразумевается конструкция, составленная из констант, переменных, знаков операций, функций, скобок. Выражение определяет порядок вычисления некоторого значения. Если это числовое значение, то такое выражение называют арифметическим.

Опишем набор операций, используемых в C, а также правила записи и вычисления выражений. Напомним, что операция, применяемая к одному операнду, называется унарной, а операция с двумя операндами — бинарной.

2. Арифметические операции

К арифметическим операциям относятся:

- вычитание или унарный минус;
+ сложение или унарный плюс;
* умножение;
/ деление;
% деление по модулю (аналог Mod в Паскале);
++ унарная операция увеличения на единицу (инкремент);
- - унарная операция уменьшения на единицу (декремент).

Все операции, кроме деления по модулю, применимы к любым числовым типам данных. Операция % применима только к целым числам. Если делимое и делитель — целые числа, то и результат — целое число. Например, значение выражения 5/3 будет равно 2, а при вычислении 1/5 получится 0.

Если хотя бы один из операндов имеет вещественный тип, то и результат будет вещественным. Например, операции 5./3, 5./3., 5/3. дадут вещественный результат 1.6666.

Операции инкремента и декремента могут применяться только к переменным и не могут — к константам и выражениям. Операция ++ увеличивает значение переменной на единицу, операция — уменьшает значение переменной на единицу. Оба знака операции могут записываться как перед операндом (префиксная форма), так и после операнда (постфиксная форма), например: ++x или x. Три следующих оператора дают один и тот же результат:

x=x+1; ++x; x++;

Различие проявляется при использовании префиксной и постфиксной форм в выражениях. Проиллюстрируем это на примерах.

Первый пример: a=3; b=2; d=a++*b++;

Результаты будут такими: a = 4, b= 3, d = 6.

Второй пример: a=3; b=2; d=++a*++b;

Результаты будут такими: a = 4, b = 3, d = 12.

При использовании постфиксной формы операции ++ и -- выполняются после того, как значение переменной было использовано в выражении, а префиксные операции — до использования. Поэтому в первом примере значение переменной d вычислялось как произведение 3 на 2, а во втором — как произведение 4 на 3.

3. Операции отношения

В C/C++ используется следующий набор операций отношения:

< меньше,

<= меньше или равно,

> больше,

>= больше или равно,

== равно,

! = не равно.

В стандарте языка С нет логического типа данных. Поэтому результатом операции отношения является целое число: если отношение истинно — то 1, если ложно — то 0. В языке программирования добавлен тип bool, значениями которого могут быть истина (true) или ложь (false).

Примеры отношений:

a<0, 101>=105, ‘a’=='A’ 'a'!='A'

Результатом второго и третьего отношений будет 0 — ложь; результат четвертого отношения равен 1 — истина; результат первого отношения зависит от значения переменной a.

4. Логические операции

Три логические операции в языке С записываются следующим образом:

! операция отрицания (НЕ),

&& конъюнкция, логическое умножение (И),

| | дизъюнкция, логическое сложение (ИЛИ).

Правила их выполнения определяются таблицей истинности.

Например, логическое выражение, соответствующее системе неравенств 0 <x< 1 в программе на С запишется в виде следующего логического выражения:

x>0 && x<1

5. Приоритеты операций

Выражения вычисляются в зависимости от приоритета операций, участвующих в выражении. Одинаковые по старшинству операции выполняются в порядке слева направо. Для изменения порядка выполнения операций в выражениях могут применяться круглые скобки.

Таблица 5. Приоритеты (ранги) операций

Ранг	Операции	Описание
1	( ) [ ] -> •	
2	! ~ + - ++ -- & * (тип) sizeof	унарные
3	* / %	мультипликативные бинарные
4	+ -	аддитивные бинарные
5	<< >>>	поразрядного сдвига
6	< <= >= >	отношения
7	= = ! =	отношения
8	&&	конъюнкция «И»
9	||	дизъюнкция «ИЛИ»
10	?:	условная
11	= *= /= %= += -= ^= |= <<= >>=	сокращенные

Пример программы с логическими операциями

Составить программу, вычисляющую истинность выражения:

a && ! b || c && ! d && (a && b)

при a=2; b=-3; c=0; d=1; main()

```
{       int a, b, c, d, rez; a=2; b=-3;
c=0; d=1; rez=a && ! b || c && ! d &&
(a && b); cout<<"\n a=2; b=-3; c=0;
d=1;";
cout<<"\n rez=a && ! b || c && ! d && (a && b); rez=>"<<rez;
}
```

Результаты работы

a=2; b=-3; c=0; d=1;

rez=a && ! b || c && ! d && (a && b); rez=> 0

Задание

На оценку «Удовлетворительно»

Вычислить значение логического выражения при a=2; b=-3; c=0; d=1;

rez=a && ! b || c && ! d && (a && b);

На оценку «Хорошо»

Определить истинность высказывания

x > y && ! (x > 0 || z > x)

1) установить порядок логических операций;

2) считая x=-1 y=1 z=-2 определить истинность высказывания.

На оценку «Отлично»

Составить условие для определения истинности следующего логического высказывания: является ли число a четным и кратным 4 и 5.

Контрольные вопросы

1. Что является результатом операции отношения?
2. Как в языке С кодируется истина и ложь? 3. Представить таблицу истинности для логических операций.

Лабораторная работа №9.
Условие попадания в заданный диапазон.

Цель работы: Получение навыков составления логических выражений.

Темы для предварительной проработки Типы данных.

Операторы ввода – вывода данных.

Теоретический материал

1. Операции и выражения

В языках программирования под выражением подразумевается конструкция, составленная из констант, переменных, знаков операций, функций, скобок. Выражение определяет порядок вычисления некоторого значения. Операция, применяемая к одному операнду, называется унарной, а операция с двумя операндами — бинарной.

2. Арифметические операции

К арифметическим операциям относятся:

- вычитание или унарный минус;
+ сложение или унарный плюс;
* умножение;
/ деление;
% деление по модулю (аналог Mod в Паскале);
++ унарная операция увеличения на единицу (инкремент);
- - унарная операция уменьшения на единицу (декремент).

Все операции, кроме деления по модулю, применимы к любым числовым типам данных. Операция % применима только к целым числам.

Операции инкремента и декремента могут применяться только к переменным и не могут — к константам и выражениям. Операция ++ увеличивает значение переменной на единицу, операция — уменьшает значение переменной на единицу.

3. Операции отношения

В С используется следующий набор операций отношения:

< меньше,
<= меньше или равно,
> больше,
>= больше или равно,
== равно,
! = не равно.

Результатом операции отношения является целое число: если отношение истинно — то 1, если ложно — то 0. Примеры отношений:

a<0, 101>=105, ‘a’=='A’ 'a'!='A'

Результатом второго и третьего отношений будет 0 — ложь; результат четвертого отношения равен 1 — истина; результат первого отношения зависит от значения переменной a.

4. Логические операции

Три логические операции в языке С записываются следующим образом:

! операция отрицания (НЕ),
&& конъюнкция, логическое умножение (И), |
| дизъюнкция, логическое сложение (ИЛИ).

По убыванию приоритета логические операции и операции отношения расположены в следующем порядке:

!
> < >= <=

= = ! =

&&

||

Пример программы попадания в заданный диапазон

Программа демонстрирует определение условий попадания заданной точки внутрь, вне и на контур указанной фигуры. Исходными данными к задаче являются координаты некоторой точки. Результатом работы программы является 1 или 0 в зависимости от результата попадания.

```
main()
{       float x,y; int rez,var;
cout<<"\n vvedi x=> ";                  cin>>x;
cout<<"\n vvedi y=> ";                  cin>>y;
cout<<"\n x y"<<x<<y;
rez=(x==-1 && (y>=-1 && y<=1)) \
|| (x==1 && (y>=-1 && y<=0)) \
 || (y==1 && (x>=-1 && x<=0)) \
 || (y==-1 && (x>=-1 && x<=1)) \
 || (x==0 && (y>=0 && y <=1)) \  ||
(y==0   &&   (x>=0   &&   x<=1));
cout<<"\n                  *****kontur
rez=>"<<rez;
rez=(x>-1 && x<1) && (y>-1 && y<1) && ! ((x>0 && x<1) \
&& (y>0 && y<1));
cout<<"\n *****wnutry rez=>"<<rez;
rez=(x<-1 || x>1) || (!(x>=-1 && x<=0 && y>=-1 && y<=1) && \
 !(x>=0 && x<=1 && y>=-1 && y<=0));
cout<<"\n *****wne figura rez=>"<<rez; }
```

Результаты работы программы

```
vvedi     x=>     -1
vvedi y=> 0
x     y          -1.000000      0.000000
*****kontur rez=> 1
*****wnutry rez=> 0
*****wne figura rez=> 0
```

Задание

На оценку «Удовлетворительно»

Написать программу, определяющей условия попадания точки, заданной своими координатами, внутрь, вне и на контур заштрихованной фигуры.

На оценку «Хорошо»

Написать программу, определяющей условия попадания точки, заданной своими координатами, внутрь, вне и на контур заштрихованной фигуры.

На оценку «Отлично»

Написать программу, определяющей условия попадания точки, заданной своими координатами, внутрь, вне и на контур заштрихованной фигуры. **Контрольные вопросы**

1. Перечислите операций по возрастания их приоритетов.
2. Сформируйте условия для проверки попадания точки, заданной своими координатами в круг единичного радиуса. 3. Какие операции называются унарными; бинарным?

Лабораторная работа №10.
Создание простейшего варианта программы-теста.

Цель: Приобретение навыков программирования ветвящихся алгоритмов. **Темы для предварительной проработки**

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Создание программ с линейной структурой.

Логические выражения.

Оператор условия.

Теоретический материал

Для программирования ветвящихся алгоритмов в языке C/C++ имеется несколько различных средств. К ним относятся операция условия ? :, условный оператор if и оператор выбора switch. *1. Оператор выбора (переключатель)* Формат оператора выбора: switch (целочисленное_выражение)

{ case константа1: список_операторов;
break; case константа2:
список_операторов; break;
……..
default:
список__операторов; }

Последняя строка (default) может отсутствовать. Оператор выбора используется для замены вложенных ветвлений и для выбора единственного варианта выполнения команд из предложенного набора. Является компактной формой задания проверок и ветвлений в программах.

Выполнение оператора switch происходит в следующем порядке:

1. вычисляется целочисленное выражение;
2. полученное значение последовательно сравнивается с константами, помещенными после служебного слова case;
3. при первом совпадении значений выполняются операторы, стоящие после двоеточия;
4. если ни с одной из констант совпадения не произошло, то выполнятся операторы после слова default.

2. Оператор прерывания

Оператор switch имеет следующую особенность выполнения: при совпадении значения вычисленного выражения с меткой, выполняется не только операторы, относящиеся к этой метке, но все нижележащие, вплоть до оператора default. Это не всегда соответствует необходимой ситуации. Для того чтобы «обойти» выполнение операторов на последующих ветвях конструкции switch, нужно принять специальные меры, используя операторы выхода или перехода. Для этих целей часто используется оператор break. Его следует писать в каждой ветви для case. После его выполнения происходит выход из всех конструкции switch на следующей за ней команду.

Пример программы с использование оператора выбора

Составить программу с одним вопросом и четырьмя вариантами ответа. Компьютер спрашивает у пользователя вариант ответа, сравнивает ответ с правильным вариантом и выдает подходящее сообщение.

```
main()
{       int     nom;            // ответ испытуемого
cout<< “\nВ каком году был основан Санкт-Петербург?\n”;
cout<< “1. 2000\n”; cout<< “2. 1700\n”; cout<< “3. 1703\n”;
cout<<“4. 1850\n”;
```

```
cout<< “Введите Ваш выбор и нажмите <Enter>”;
cout<<”-> “; cin>>nom; switch (nom)
{case 1: cout<< “Вы ответили неправльно”; break;
case 2: cout<< “Вы ответили неправльно”; break;
case 3: cout<<“Вы ответили правльно”; break;
case 4: cout<<“Вы ответили неправльно”; break;
default: cout<<“Такого варианта ответа нет”;
}
}
```

Задание

На оценку «Удовлетворительно»

Составить программу-тест по своей тематике с пятью вопросами. Пройти тест и выдать оценку в диапазоне от 0 до 5. *На оценку «Хорошо»*

В зависимости от полученной оценки при прохождении теста, вывести подходящее сообщение: «Неудовлетворительно», «Удовлетворительно», «Хорошо», «Отлично». *На оценку «Отлично»*

Для каждого правильного ответа на тест вывести сообщение «Молодец», для каждого неправильного ответа вывести сообщение «Плохо».

Контрольные вопросы

1. Назовите основное назначение оператора switch.
2. Запишите синтаксис оператора switch. 3. Объясните назначение оператора break.

Лабораторная работа №11.
Программа на базе алгоритма с полным ветвлением.

Цель: Приобретение навыков программирования ветвящихся алгоритмов.

Темы для предварительной проработки

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Создание программ с линейной структурой.

Логические операции и операции отношения.

Теоретический материал

1. Условный оператор

Формат условного оператора следующий:

if (выражение) оператор1; else оператор2;

Это полная форма оператора, программирующая структуру полного ветвления. Обычно выражение — это некоторое условие, содержащее операции отношения и логические операции. Значение выражения приводится к целому и интерпретируется в соответствии с правилом: равно нулю — ложь, не равно нулю — истина. Если выражение истинно, выполняется оператор1, если ложно — оператор2.

Необходимо обратить внимание на следующие особенности синтаксиса условного оператора:

- выражение записывается в круглых скобках;
- точка с запятой после оператора1 ставится обязательно.

2. Неполная форма условного оператора

Возможно использование неполной формы условного оператора. Используется, когда отсутствую действия в случае истинности логического выражения.

if (выражение) оператор;

Пример. Упорядочить по возрастанию значения в двух переменных a, b:

```
if(a>b)
{
c=a;          a=b;          b=c;
}
cout«"a="<<a<<"b="<<b;
```

В данном примере использован составной оператор — последовательность операторов, заключенная в фигурные скобки.

Обратите внимание на то, что перед закрывающей фигурной скобкой точку с запятой надо ставить обязательно, а после скобки точка с запятой не ставится.

Пример программы с применением полного и неполного ветвлений

Найти большее значение из двух переменных a и b. Вариант 1. С использованием полной ветви

```
main()
{       float a, b, max;
cout<<"\n vvedi a=> ";              cin>>a;
cout<<"\n vvedi b=> ");     cin>>b;
if (a>b) max=a;
else
max=b;
cout<<"\n max="<<max;
}
```

Вариант 2. С использованием неполной ветви

```
main() {
float a, b, max;
cout<<"\n vvedi a=> ";              cin>>a;
cout<<"\n vvedi b=> ";              cin>>b; max=b;
if          (a>b)
max=a;
cout<<"\n max="<<max;
}
```

Задание

На оценку «Удовлетворительно»

Ввести три числа и найти наибольшее из них.

Пример: Введите три числа: 4 15 9

Наибольшее число 15

На оценку «Хорошо»

Составить программу упорядочения по возрастанию значений в трех переменных, вводимых пользователем с клавиатуры.

На оценку «Отлично»

Ввести номер месяца и вывести название времени года.

Пример: Введите номер месяца: 4

Весна

Контрольные вопросы 1. С какой целью в программах используется оператор условия if?

2. Укажите синтаксис полного ветвления.
3. Укажите синтаксис неполного ветвления.

Лабораторная работа №12.
Программа решения квадратного уравнения.

Цель: Получение навыков в работе с разветвляющимися алгоритмами.

Темы для предварительной проработки Арифметические операторы С++ .

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Теоретический материал

Для программирования ветвящихся алгоритмов в языке C/C++ и имеется несколько различных средств. К ним относятся рассмотренная выше операция условия ? :, условный оператор if и оператор выбора switch.

1. Условный оператор

Формат условного оператора следующий: **if (выражение) оператор1; else оператор2;**

Это полная форма оператора, программирующая структуру полного ветвления. Обычно выражение — это некоторое условие, содержащее операции отношения и логические операции. Значение выражения приводится к целому и интерпретируется в соответствии с правилом: равно нулю — ложь, не равно нулю — истина. Если выражение истинно, выполняется оператор1, если ложно — оператор2.

Необходимо обратить внимание на следующие особенности синтаксиса условного оператора:

- выражение записывается в круглых скобках;
- точка с запятой после оператора 1 ставится обязательно;

Вот пример использования полной формы условного оператора для нахождения большего значения из двух переменных a и b:

```
if(a>b) max=a;
else        max=b;
```

2. Вложенные условия

Вложенные ветвления возникают в программах, когда в качестве действий по ветви истина или лжи возникает еще одна ветвь. Вложенные ветви могут быть полными (с else) а могут быть неполными (без else). Рассмотрим примеры программирования вложенных ветвящихся структур.

Требуется вычислить функцию sign(x) — знак x, которая определена следующим образом:

Пример 1. Алгоритм с полными вложенными ветвлениями:

```
if (x<=0) if(x==0) y=0; else y=-1;  else y=1;
```

Пример 2.

Алгоритм с неполным вложенным ветвлением:

```
y=1;
if(x<=0)
if(x==0)
y=0;
else y=-
l;
```

Пример программы решения квадратного уравнения

```
main()
```

```
{       float a ,b, c;                          // коэффициенты уравнения float
x1, x2;                                 // корни уравнения
float d;                                // дискриминант cout<<"\n*
Решение квадратного уравнения *\n"; cout<<"Введите в одной
строке значения коэффициентов"; cout<<" и нажмите
<Enter>";
cout<<"-> ";
cin>>a>>b>>c;                           // ввод коэффициентов d
= b*b - 4*a*c;                          // дискриминант
if (d < 0)        cout<<"Уравнение не имеет решения\n";
else  if (d > 0){ x1 = (-b + sqrt(d))/(2*a); x2 = (-b -
sqrt(d))/(2*a);
cout<<"Корни уравнения: x1=”<<x1<<”x2=”<<x2;
}
else     {
x1 = -b/(2*a); x2 = x1; cout<<"Корни уравнения:
x1=”<<x1<<”x2=”<<x2;
}
}
```

Задание

На оценку «Удовлетворительно»

Составить блок-схему алгоритма решения квадратного уравнения. Проверить работу программы на различных исходных данных.

На оценку «Хорошо»

Добавить проверку условия коэффициента a, равного. При положительной проверке решить линейное уравнение.

На оценку «Отлично»

Рассмотреть случай равенства 0 коэффициентов b и c, входящих в квадратное уравнение.

Контрольные вопросы

1. Когда в программах возникает необходимость использования вложенных ветвей?
2. Перечислите разновидности вложенных ветвлений и примеры блок-схем для каждого случая. 3. Объясните правило использования фигурных скобок во вложенных ветвях.

Лабораторная работа №13.
Программирование вложенных ветвлений.

Цель: Приобретение навыков программирования с вложением ветвлений.

Темы для предварительной проработки

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Создание программ с ветвлением.

Теоретический материал

1. Вложенные if-инструкции

Вложенные ветвления образуются в том случае, когда в качестве элемента инструкция (см. полный формат записи) используется другая if-инструкция. Вложенные if-инструкции очень популярны в программировании. Главное здесь — помнить, что else-инструкция всегда относится к

ближайшей if-инструкции, которая находится внутри того же программного блока, но еще не связана ни с какой другой else-инструкцией.

Продемонстрируем использование вложенных инструкций с помощью программы "Угадай магическое число".

```
main()
{int magic;                 //магическое число
int guess;                  //вариант пользователя
magic = rand();             // Получаем случайное число.
cout << "Введите свой вариант магического числа:";
cin >>guess; if (guess == magic)
{cout << "** Правильно **\n";
cout << magic << " и есть то самое магическое число. \n";
} else { cout <<"Очень жаль, но вы ошиблись."; if
(guess > magic) cout <<" Ваш вариант превышает
магическое число.\n"; else cout << " Ваш вариант
меньше магического числа.\n"; }
```

2. Конструкция if -else-if

Очень распространенной в программировании конструкцией, в основе которой лежит сложенная if-инструкция, является "лестница" if-else-if. Ее можно представить в следующем виде.

```
if (условие)
инструкция;
else
if(условие)
инструкция;
else          if
(условие)
инструкция;
else
инструкция;
```

Здесь под элементом условие понимается условное выражение. Условные выражения вычисляются сверху вниз. Как только в какой-нибудь ветви обнаружится истинный результат, будет выполнена инструкция, связанная с этой ветвью, а вся остальная "лестница" опускается. Работа if -else-if-"лестницы" демонстрируется в следующей программе.

```
main()
{int x;
x=rand()%10;
if (x==l)          cout << "x равен единице.\n";
else if(x==2) cout « "x равен двум.\n";
else
if(x==3)
cout << "x равен трем.\n";
else
if(x==4)
cout << "x равен четырем.\n"; else cout << "x не
попадает в диапазон от 1 до 4.\n"; }
```

Пример программы с вложенными ветвлениями

Пример демонстрирует справедливую шкалу подоходного налога. Здесь d – размер зарплаты. pn – размер подоходного налога. Например, если d<20000, pn=d*0.12, если d>20000, pn=2400+(d-20000)*0.13 и т.д.

```
main()
{       float d, pn;
cout<<"\nvvedi number
"; cin>>d; if (d>100000)
pn=20400+(d-100000)*0.35;
else if(d>=80000)
pn=14000+(d-80000)*0.3;
else if(d>=60000)
pn=9400+(d-60000)*0.25;
else if(d>=40000)
pn=5400+(d-
40000)*0.2; else
if(d>=20000) pn=2400+
(d-20000)*0.13;
else
pn=d*0.12;
cout << "\n pn = "<<pn;
}
```

Задание

На оценку «Удовлетворительно»

Даны три числа a, b, c. Определить, какое из них равно d. Если ни одно не равно d, то найти max(d-a, d-b, d-c).

На оценку «Хорошо»

Составить программу, реализующую эпизод применения компьютера в книжном магазине. Компьютер запрашивает стоимость книг, сумму денег, внесенную покупателем; если сдачи не требуется, печатает на экране «спасибо»; если денег внесено больше, чем необходимо, то печатает «возьмите сдачу» и указывает сумму сдачи; если денег недостаточно, то печатает сообщение об этом и указывает размер недостающей суммы.

На оценку «Отлично»

Плата за сотовый телефон взимается по следующему тарифу: за первые 50 минут разговора в месяц – по 1 р., за следующие 30 минут – по 80 коп., следующие 20 минут обойдутся абоненту по 50 коп., а каждая последующая минута стоит всего 30 коп. Известно, сколько минут в месяц абонент наговорил. Определите, какую сумму он должен заплатить.

Контрольные вопросы

1. Когда в программах возникает вложенная ветвь?
2. Перечислить правила работы конструкции «лестница».
3. Сформулировать правило взаимодействия if и else.

Лабораторная работа №14.
Программа с применением конструкции выбора.

Цель: Приобретение навыков программирования ветвящихся алгоритмов.

Темы для предварительной проработки

Объявление и инициализация переменных.

Функции стандартного ввода и вывода.

Создание программ с линейной структурой.

Логические выражения.

Оператор условия.

Теоретический материал

Для программирования ветвящихся алгоритмов в языке C/C++ имеется несколько различных средств. К ним относятся операция условия ? :, условный оператор if и оператор выбора switch. *1. Оператор выбора (переключатель)* Формат оператора выбора:

```
switch (целочисленное_выражение)
{ case константа1: список_операторов;
break; case константа2:
список_операторов; break;
……..
default:
список__операторов; }
```

Последняя строка (default) может отсутствовать. Оператор выбора используется для замены вложенных ветвлений и для выбора единственного варианта выполнения команд из предложенного набора. Является компактной формой задания проверок и ветвлений в программах.

Выполнение оператора switch происходит в следующем порядке:

1. вычисляется целочисленное выражение;
2. полученное значение последовательно сравнивается с константами, помещенными после служебного слова case;
3. при первом совпадении значений выполняются операторы, стоящие после двоеточия;
4. если ни с одной из констант совпадения не произошло, то выполнятся операторы после слова default.

2. Особенности команды множественного выбора

1. Для switch нужен отдельный блок.
2. Тип выражения может быть целым или символьным, но не вещественным.
3. Константы в case должны иметь тот же тип, что и выражение.
4. Если действие подходит под несколько Констант, то следует несколько case подряд. Например:

```
switch  (выражение)
{
case 1: case 2: case 5: Действия1;
case 3: case 4: Действия2; }
```

5. Константы в пределах одного switch должны быть уникальны, т.е. не должны повторяться. Нарушение этого приводит к неоднозначности выбора нужного оператора.
6. Если для подходящей Константы требуется выполнить не один оператор, а несколько, то в блок их объединять не обязательно, т.к. действия case распространяется до конца, пока не встретиться конец switch }, или команда break.
7. Команда switch является частным случаем команды if. Когда нельзя применять команду switch:

- выражение вещественное;
- выражение проверяется в некотором диапазоне даже для целых, например: $1 \le a \le 100$. В этом случае легче написать if (a>=1&&a<=100) чем выписывать 100 case.
- выражение проверяется на точное совпадение со значением констант в case, сравнения на <, > в switch не допускаются;
- проверка на строгое равенство проверяется только для одного значения – вычисленного значения выражения, switch не подходит для проверки условий для нескольких величин, например if (x>10 && y<5) с помощью switch задать такое условие не удастся.

3. Оператор прерывания

Оператор switch имеет следующую особенность выполнения: при совпадении значения вычисленного выражения с меткой, выполняется не только операторы, относящиеся к этой метке, но все нижележащие, вплоть до оператора default. Это не всегда соответствует необходимой ситуации. Для того чтобы «обойти» выполнение операторов на последующих ветвях конструкции switch, нужно принять специальные меры, используя операторы выхода или перехода. Для этих целей часто используется оператор break. Его следует писать в каждой ветви для case. После его выполнения происходит выход из всех конструкции switch на следующей за ней команду.

Пример программы с использование оператора выбора Программа-калькулятор.

Вводится выражение вида -5.45*3.56

Программа должна вычислить и вывести результат выражения.

В общем виде выражение выглядит так: операнд1 операция операнд2

Операндами являются вещественные числа, в качестве операции можно использовать четыре основных арифметических действия: + - * /, которые вводятся в программу как символы.

```
main()
{       float a, b, rez;
char oper;
cin>>a>>oper>>b;
switch (oper)
{ case '+': rez=a+b;
break; case '-': rez=a-b;
break; case '*': rez=a*b;
break; case '/': rez=a/b;
break;
}
cout<<" Результат="<<rez<,endl;
```

Задание

На оценку «Удовлетворительно»

Вводится номер месяца. Выдать название времени года.

На оценку «Хорошо»

Вводится количество лет К – целое число от 1 до 99. Вывести фразу «Мне … лет», где надо заменить слово «лет» на подходящее по смыслу слова «год» и «года».

Заметим закономерность, как от значения К зависит слово.

а) если число заканчивается цифрой 1, то слово «год» кроме 11 лет;

б) если число К заканчивается цифрами 2, 3, 4 , то «года» кроме 12, 13, 14 лет;

в) если число К заканчивается цифрой 0, 5..9, то «лет».

На оценку «Отлично»

Вводится дата в виде: день (d), месяц (m), год (g). Выдать название для недели, на которую приходится введенная дата.

<День недели> = <остаток от деления X> на 7, где

X = [2.6*m - 0.2]+d+[y / 4]+y+[c / 4] - 2*c, где [] означают целую часть числа.

c – две старшие цифры года;

y – две младшие цифры года.

При использовании этой формулы следует учесть два обстоятельства:

1) формула верна для григорианского календаря нового стиля (от 1582 до 4902);

2) месяц следует предварительно преобразовать так, как если бы начало года приходилось на 1 марта. Иными словами, март в этой формуле имеет порядковый номер 1,

апрель – 2, …, январь – 11 и февраль – 12, причём январь и февраль следует отнести к предыдущему году. Например, для

1 февраля 1991 года номер месяца должен быть равен 12, а год 1990, в то время как для 31 декабря 1991 года номера месяца – 10, а год – 1991. Результат вычисления даётся в виде целого числа в диапазоне от 0 до 6, причём 0 соответствует воскресенью.

Контрольные вопросы

1. Назовите основное назначение оператора switch.
2. Запишите синтаксис оператора switch. 3. Объясните назначение оператора break.

Лабораторная работа №15.
Программа вычисления сумм и количества, поиск максимального и минимального значений.

Цель: Получение навыков в работе с арифметическими выражениями, циклическими инструкциями.

Темы для предварительной проработки Арифметические операторы C++.

Функции стандартного ввода и вывода.

Оператор ветвления.

Теоретический материал

1. Цикл с параметром

Формат оператора цикла с параметром:

for (выражение_1; выражение_2; выражение_3) оператор;

Выражение_1 выполняется только один раз в начале цикла. Обычно оно определяет начальное значение параметра цикла (инициализирует параметр цикла). Выражение_2 — это условие выполнения цикла. Выражение_3 обычно определяет изменение параметра цикла. Оператор — тело цикла, которое может быть простым или составным. В последнем случае используются фигурные скобки.

Алгоритм выполнения цикла for представлен на блок-схеме:

После вычисления выражения_3 происходит возврат к вычислению выражения_2 — проверке условия повторения цикла.

Рассмотрим некоторые особенности арифметического цикла на примере вычисления факториала числа. С помощью цикла for нахождение N! можно организовать следующим образом:

```
F=l;
for (i=l; i<=N; i++)     F=F*i;
```

Используя операцию «запятая», можно в выражение_1 внести инициализацию значений сразу нескольких переменных:

```
for(F=l, i=l; i<=N; i++)                    F=F*i;
```

Некоторых элементов в операторе for может не быть, однако разделяющие их точки с запятой обязательно должны присутствовать. В следующем примере инициализирующая часть вынесена из оператора for:

```
F=l;
i=l;
for (; i<=N; i++)                    F=F*i;
```

Ниже показан еще один вариант вычисления N!. В нем на месте тела цикла находится пустой оператор, а вычислительная часть внесена в выражение_3.

```
for(F=l, i=l; i<=N; F=F*i, i++);
```

Пример программы с арифметическим циклом

Рассмотрим программу, определяющую сумму, а так же наибольшее и наименьшее значения среди последовательности 10 случайных величин, задаваемых с помощью датчика случайных чисел.

```
main()
{       int i, n, min, max, s;
randomize();
max=0;
min=100; s=0;
for(i=1; i<=10; i++)
{
n=random(100)
;    cout<<n<<"
";    s+=n;    if
(n>max)
max=n;        if
(n<min)
min=n;
}
cout<<"\n ****** max="<<max<<" min="<<min<<” s=”<<s;
}
```

Результаты работы программы:

12 36 78 21 20 37 89 90 70 52

****** max=90 min=12 s=505

Задание

На оценку «Удовлетворительно»

Разработать программу, определяющую сумму, а так же наибольшее и наименьшее значения среди последовательности 10 случайных величин, диапазон которых мог задавать пользователь.

На оценку «Хорошо»

Разработать программу, определяющую сумму, а так же наибольшее и наименьшее значения среди последовательности случайных величин, количество которых мог задавать пользователь.

На оценку «Отлично»

Разработать программу, определяющую сумму, а так же наибольшее и наименьшее значения среди последовательности случайных величин, а так же позиции, в которых достигаются максимальное и минимальное значения.

Контрольные вопросы

1. Дать определение циклическому алгоритму.
2. Запиши команду арифметического цикла с особенностями применения. 3.
   Составь алгоритм вычисления суммы заданного количества величин.

Лабораторная работа №16.
Программирование прикладных задач с применением циклических алгоритмов.

Цель: Получение навыков при разработке прикладных программ с применением циклических конструкций.

Темы для предварительной проработки Арифметические операторы С++.

Функции стандартного ввода и вывода.

Оператор ветвления. **Теоретический материал** *1. Числа Фибоначчи*

Известна следующая задача о кроликах. Поместили пару кроликов в некоем месте, огороженном со всех сторон стеной, чтобы узнать, сколько пар кроликов родится при этом в течение года, если природа кроликов такова, что через месяц пара кроликов производит на свет другую пару, а рождают кролики со второго месяца после своего рождения.

Ясно, что если считать первую пару кроликов новорожденными, то на второй месяц мы будем по прежнему иметь одну пару; на 3-й месяц – 1 + 1 = 2; на 4-й – 2 + 1 = 3 пары (ибо из двух имеющихся пар потомство дает лишь одна пара); на 5-й месяц – 3 + 2 = 5 пар (лишь 2 родившиеся на 3-й месяц пары дадут потомство на 5-й месяц); на 6-й месяц – 5 + 3 = 8 пар (ибо потомство дадут только те пары, которые родились на 4-м месяце) и т. д.

Если обозначить число пар кроликов, имеющихся на n-м месяце через Fn , то F1 = 1, F2 = 1, F3 = 2, F4 = 3, F5 = 5, F6 = 8, F7 = 13, F8 = 21 и т. д., причем образование этих чисел регулируется общим законом: $F_n = F_{n-1} + F_{n-2}$ при всех n>2, ведь число пар кроликов на n-м месяце равно числу F_{n-1} пар кроликов на предшествующем месяце плюс число вновь родившихся пар, которое совпадает с числом F_{n-2} пар кроликов, родившихся на (n – 2)-м месяце (ибо лишь эти пары кроликов дают потомство).

Французский математик Люка впервые назвал числовую последовательность 1, 1, 2, 3, 5, 8, 13…, которую назвал числами Фибоначчи и открыл не менее важную последовательность: 2, 1, 3, 4, 7, 11…,которая связана с золотой пропорцией. Отношение соседних чисел Люка по мере удаления от начала последовательности в пределе стремится к золотой пропорции.

Последовательности чисел Фибоначчи F(n) = 1, 1, 2, 3, 5, 8, 13,… и чисел Люка: L(n) = 2, 1, 3, 4, 7, 11,… ученые все чаще встречают во многих явлениях окружающего мира. *2. Рекуррентные соотношения*

Рекуррентными называются соотношения, в которых очередной член последовательности выражен через один или несколько предыдущих (от латинского "recurrere" – возвращаться").

Пример: формулы для нахождения n-го элемента

$a_1 = 6$, $a_i = a_{i-1} + 3$ 6, 9, 12, 15, 18, 21, 24, 27, 31 a_1 = 1, $a_i = a_{i-1} + 10$ 1, 11, 21, 31, 41, 51, 61, 71,81 a_1 = 3, $a_i = a_{i-1} * 2 – 1$ 5, 9, 17, 33, 65, 129, 257, 513

Последовательности чисел Фибоначчи F(n) = 1, 1, 2, 3, 5, 8, 13,… является рекуррентной последовательность, для которой выполняются условия:

$F_1=F_2=1$;

$F_n=F_{n-1}+F_{n-2}$

Каждый член такой последовательности, кроме первых двух, зависит от значения двух предыдущих членов последовательности.

3.Экологическая задача

В результате сброса промышленных стоков возрос уровень загрязнения реки. Каким он будет через сутки, двое и т.д. и когда уровень загрязнения станет допустимым, если известно, что за сутки он уменьшается в определенное количество раз?

Обозначим: С0 – начальная концентрация вредной примеси.

Сдоп – предельно допустимая концентрация вредной примеси.

К – коэффициент суточного уменьшения концентрации вредного вещества.

Таблица 6. Исходные данные к экологической задаче

Вещество	C0 (мг/л)	Сдоп (мг/л)	K
Свинец	5	0.03	1.12
Мышьяк	1.5	0.05	1.05
Фтор	0.2	0.05	1.01

Математическая формулировка задачи:

C1 = C0 / K

C2 = C1 / K = C0 / K2

C3 = C2 / K = C1 / K3

……

Cn = Cn-1 / K = C0 / Kn или

Cn = C0 / Kn или

Cn = Cn-1 / K

Анализируем последнюю формулу – состояние загрязнения реки в текущий день зависит от состояния загрязнения в предыдущий день, т.е. имеем рекуррентное соотношение.

Пример программы с рекуррентными последовательностями

Разработаем программу для расчёта содержания в реке свинца. Обозначим предельно допустимое значение концентрации вредной примеси cdop, начальное значение концентрации c0 , коэффициент уменьшения загрязнения в сутки k, i – число суток. Для создания алгоритма представим рекуррентную формулу в виде: c=cpred / k.

```
main()
{
float c0, cdop, k, c, cpred; int
i;
c0=5.0;  cdop=0.03;  k=1.12;      //Из  таблицы
cpred=c0; for (i=1;  ;i++)
{
c=cpred/k;
if (c<=cdop)            break;
cpred=c; } cout<<i; }
```

Результат работы программы:

46

Задание

На оценку «Удовлетворительно»

Разработаем программу для расчёта содержания в реке мышьяка.

На оценку «Хорошо»

Разработаем программу для расчёта содержания в реке фтора.

На оценку «Отлично»

Разработаем программу для расчёта содержания в реке условного вещества, показатели которого вводятся пользователем.

Контрольные вопросы

1. Какие соотношения называются рекуррентными?
2. Приведите первых 10 чисел Фибоначчи, используя рекуррентную формулу.

Лабораторная работа №17.
Итерационные циклы. Поиск корней нелинейного уравнения методом деления пополам.

Цель: Получение навыков в работе с циклическими алгоритмами. Реализация на их основе математических расчетов.

Темы для предварительной проработки Арифметические операторы С++.

Функции стандартного ввода и вывода.

Оператор ветвления.

Теоретический материал

1. Основные понятия для циклов

Цикл - это алгоритмическая конструкция, в которой в зависимости от условия повторяется определённая последовательность действий. Для организации циклов можно применять условия, т.к. циклический алгоритм является частным случаем разветвляющегося. **Тело цикла** - это повторяющаяся последовательность действий (блок инструкций).

Цикл, для которого нельзя указать число повторений, и проверка окончания которого происходит по достижению нужного условия, называется **итерационным**. Цикл выполняется до тех пор, пока выражение отлично от нуля, т.е. заключенное в нем условие цикла истинно. Выход из цикла происходит после того, как значение выражения станет ложным, иными словами равным нулю.

2. Цикл с предусловием

В цикле с предусловием, называемом еще циклом “пока”, сначала проверяется условие, а затем выполняются действия.

Цикл с предусловием может не выполниться ни разу.

Формат оператора цикла с предусловием:

while (выражение) оператор;

3. Цикл с постусловием

В цикле с постусловием сначала выполняется действие, а лишь потом проверяется условие. Цикл “до” функционирует следующим образом: до тех пор, пока условие не выполнится, выполняется тело цикла. Характерно, что тело цикла в цикле ”до” выполняется хотя бы один раз.

Формат оператора цикла с постусловием:

do оператор while (выражение);

Цикл с предусловием. Формат оператора цикла с предусловием:

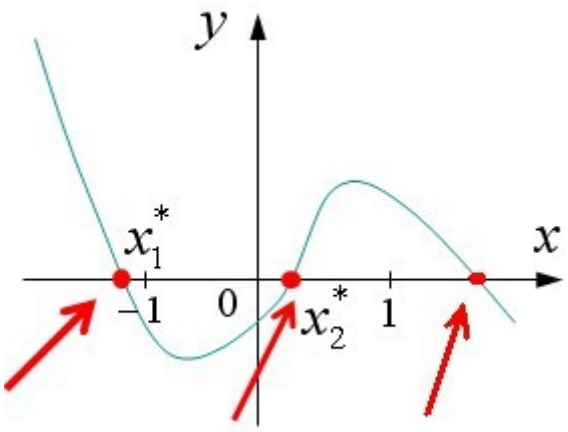
while (выражение) оператор;

Цикл повторяет свое выполнение, пока значение выражения отлично от нуля, т.е. заключенное в нем условие цикла истинно.

3.4. Решение нелинейного уравнения методом деления пополам

Пусть требуется решить нелинейное уравнение вида f(x)=0 на заданном отрезке [a; b] с необходимой точностью ε>0.

Решить уравнение – это найти его корень (корни). Корнем называется такое значение x*, подставив которое в f(x) получается ноль, т.е. f (x*)=0



Графически корень – это точка пересечения графика функции y=f (x) с осью ox.

Узнаем, есть ли на отрезке [a,b] решение. Для этого надо провести исследование, как ведет себя функция f(x) на концах отрезка [a; b].

Рисунок 1 – Корни нелинейного уравнения

-нет решения -нет решения -есть решение

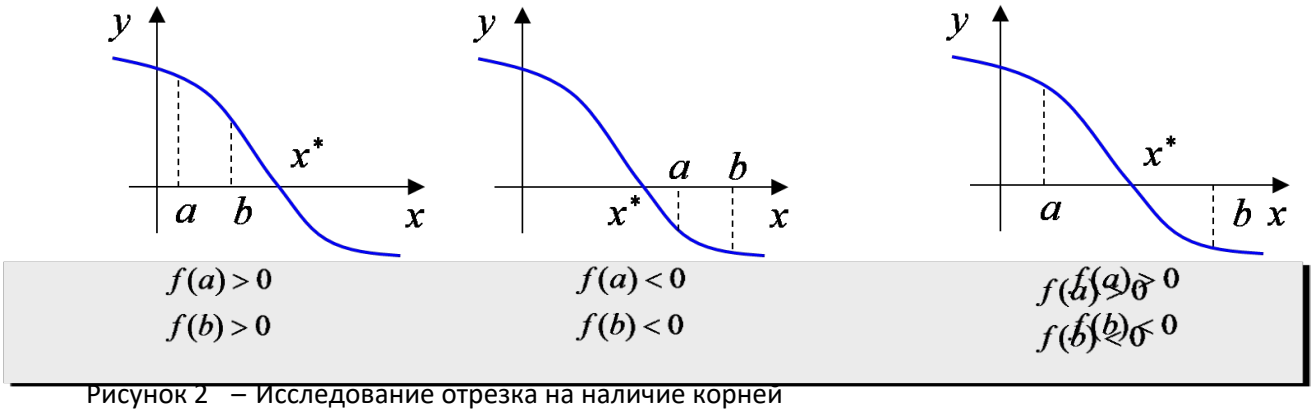


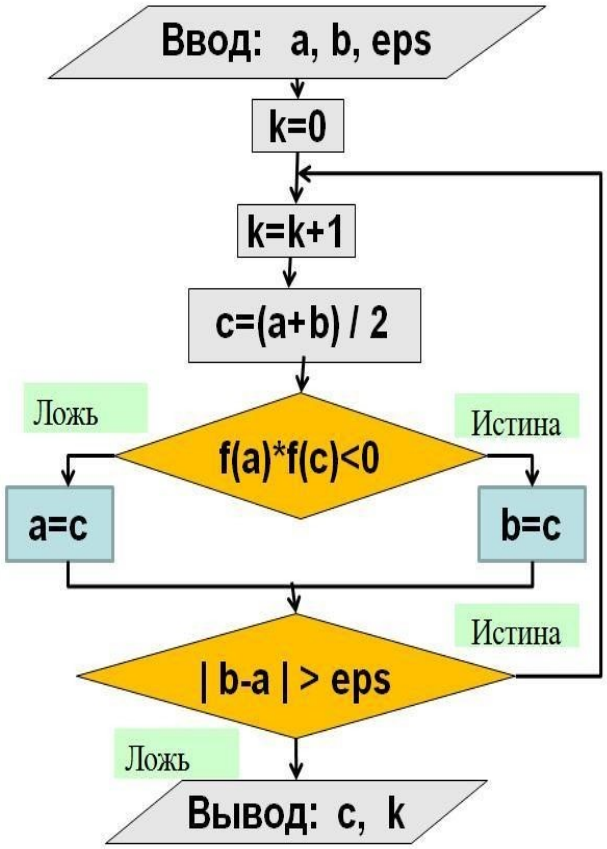
Рисунок 2 – Исследование отрезка на наличие корней

Если непрерывная монотонная функция f (x) имеет разные знаки на концах отрезка [a, b], то она имеет единственный корень на данном отрезке.

$f(a) \times f(b) < 0$ Условие существования единственного корня на [a; b].

Для решения нелинейного уравнения можно использовать метод деления пополам, относящийся к дисциплине «Численные методы».

Алгоритм метода деления пополам



1. Найти середину отрезка [a,b]:
   c = (a + b) / 2;
2. Если f(c)*f(a)<0, сдвинуть правую границу интервала b = c;
3. Если f(c)*f(a)≥ 0, сдвинуть левую границу интервала a = c;
4. Повторять шаги 1-3, пока не будет | b - a | ≤ ε.

Рисунок 3 - Блок-схема метода деления пополам

Пример программы для решения нелинейного уравнения методом деления пополам

Фрагмент программного кода для нелинейного уравнения f (x)=cos(x)+sin(x).

```
main()
{       float a, b, c, fa, fb, fc, eps;
cout<<”Введите границы отрезка ”;
cin>>a>>b; cout<<”Введите точность
расчетов ”; cin>>eps; do
{
c = (a+b)/2;                    //Середина отрезка
fa = cos(a)+sin(b);             //Значения функции в контрольных точках
fb  =  cos(b)+sin(b);
fc = cos(c)+sin(c)
if (fa*fc<0)            b=c;    //Условие выбора корня
else                    a=c;
}
while ( fabs(b-a)>eps);                 //Условие продолжения расчетов cout<<”Корень
уравнения = ”<<c<<endl;
}
```

Задание

На оценку «Удовлетворительно»

Написать программу, вычисляющую корень нелинейного уравнения методом деления пополам на заданном отрезке с указанной точностью вычисления.

На оценку «Хорошо»

Дополнительно определить количество совершенных делений. Сделать вывод, как количество делений зависит от точности расчетов.

На оценку «Отлично»

Все расчеты отобразить в виде таблицы, в которой в пошаговом режиме представить промежуточные и конечные вычисления.

Контрольные вопросы

1. Какие циклы называются итерационными?
2. Чем отличаются итерационные циклы с предусловием и постусловием?
3. Сформулируйте алгоритм работы итерационного цикла.
4. Что называется корнем нелинейного уравнения?
5. Опишите алгоритм метода деления пополам для вычисления корня нелинейного уравнения.

Лабораторная работа №18.

Вычисление рядов с заданной степенью точности.

Цель: Получение навыков в работе с циклическими алгоритмами.

Темы для предварительной проработки Арифметические операторы С++.

Функции стандартного ввода и вывода.

Оператор ветвления.

Теоретический материал

1. Основные понятия для циклов

Цикл - это алгоритмическая конструкция, в которой в зависимости от условия повторяется определённая последовательность действий. Для организации циклов можно применять условия, т.к. циклический алгоритм является частным случаем разветвляющегося. **Тело цикла** - это повторяющаяся последовательность действий (блок инструкций).

Цикл, для которого нельзя указать число повторений, и проверка окончания которого происходит по достижению нужного условия, называется **итерационным**. Цикл выполняется до тех пор, пока выражение отлично от нуля, т.е. заключенное в нем условие цикла истинно. Выход из цикла происходит после того, как значение выражения станет ложным, иными словами равным нулю.

2. Цикл с предусловием

В цикле с предусловием, называемом еще циклом “пока”, сначала проверяется условие, а затем выполняются действия.

Цикл с предусловием может не выполниться ни разу.

Формат оператора цикла с предусловием:

while (выражение) оператор;

3.3. Цикл с постусловием

В цикле с постусловием, называемом циклом “до”, наоборот: сначала выполняется действие, а лишь потом проверяется условие.

Цикл “до” функционирует следующим образом: до тех пор, пока условие не выполнится, выполняется тело цикла. Характерно, что тело цикла в цикле ”до” выполняется хотя бы один раз.

Формат оператора цикла с постусловием:

do оператор while (выражение);

Цикл выполняется до тех пор, пока выражение

Пример программы вычисления рядов с заданной точностью

Вычислить сумму гармонического ряда: 1+ 1/2+1/3+ с заданной точностью расчетов е.

```
#include <iostream.h> #include <limits.h>
main()
{int n=l; double S=0, eps;
cout«"Введите Точность:
" ; cin>>eps;
while(1.0/n>eps && n<INT_MAX)
 {
S+=l./n;
n+ + ;
}
cout<<"\nСумма="<<S;
}
```

Файл limits.h, подключаемый препроцессором, содержит определения предельных констант для целых типов данных. В частности, константа с именем int_max равна максимальному значению типа int в данной реализации компилятора. Если для типа int используется двухбайтовое

представление, то INT_MAX=32767. В этом же заголовочном файле определены и другие константы: INT_MIN=-327 68; LONG_MAX=2147483647 и т.д.

Задание

На оценку «Удовлетворительно»

Написать программу, вычисляющую все члены числового ряда, значение которых превышает 10-5.

На оценку «Хорошо»

Даны числовой ряд и некоторое число ε. Найти сумму тех членов ряда, модуль которых больше или равен заданному ε. Общий член ряда имеет вид:

$a_n=10^n/n!$

На оценку «Отлично»

Найти наименьший номер члена последовательности, $a_n=10^n/n!$ для которого выполняется условие: $|a_n — a_{n-1}| < ε$. Вывести на экран этот номер и все элементы a_i, где i = 1, 2,..., n.

Контрольные вопросы

1. Какие циклы называются итерационными?
2. Чем отличаются итерационные циклы с предусловием и постусловием?
3. Сформулируйте алгоритм работы итерационного цикла.

Лабораторная работа №19.
Программа составления таблицы функции на заданном интервале.

Цель: Получение навыков в работе с циклическими алгоритмами

Темы для предварительной проработки Арифметические операторы С++.

Функции стандартного ввода и вывода.

Оператор ветвления. **Теоретический материал**

1. Цикл с параметром

Формат оператора цикла с параметром:

for (выражение_1; выражение_2; выражение_3) оператор;

Цикл с параметром используется в программах, когда известно количество совершаемых повторений. Выражение_1 выполняется только один раз в начале цикла. Обычно оно определяет начальное значение параметра цикла (инициализирует параметр цикла). Выражение_2 — это условие выполнения цикла. Выражение_3 обычно определяет изменение параметра цикла. Оператор — тело цикла, которое может быть простым или составным. В последнем случае используются фигурные скобки.

После вычисления выражения_3 происходит возврат к вычислению выражения_2 — проверке условия повторения цикла.

Используя операцию «запятая», можно в выражение_1 внести инициализацию значений сразу нескольких переменных.

Некоторых элементов в операторе for может не быть, однако разделяющие их точки с запятой обязательно должны присутствовать.

Пример программы табулирования функции

Составить таблицу значений функции y=|x| в диапазоне изменения аргумента от -4 до 4 с шагом 0.5.

```
#define  LB -4          //нижняя граница диапазона изменения аргумента
#define  HB  4          //верхняя граница диапазона изменения аргумента
#define  DX  0.5              //приращение аргумента
main()
{       float x, y;             //аргумент и значение функции
int n;                  //кол-во точек
int i;                          //счетчик циклов
cout<<"\nТаблица значений функции y=|x| \n";
n = (HB - LB)/DX + 1; x = LB;
for (i = 1; i <= n; i++)
{ y = fabs(x);
cout<< x<<
y;
x += DX;
}
}
```

Задание

На оценку «Удовлетворительно»

Напишите программу, составляющую таблицу значений функции y=sin(x)-cos(x) в диапазоне изменения аргумента от -4 до 4 с шагом 0.5.

На оценку «Хорошо»

Напишите программу, составляющую таблицу значений функции y=sin(x)-cos(x). Диапазон изменения аргумента, а так же шаг его изменения вводится пользователем.

На оценку «Отлично»

В построенной таблице значений функции определить точки экстремума.

Контрольные вопросы

1. Когда в программах используется цикл с параметром?
2. Укажите правила синтаксиса команды for.

Лабораторная работа №20.
Программирование циклических алгоритмов при решении прикладных задач.

Цель: Получение навыков работы с различными видами циклических алгоритмов.

Темы для предварительной проработки Арифметические операторы С++.

Функции стандартного ввода и вывода.

Оператор ветвления.

Теоретический материал *1.*

Понятие цикла

Под циклом понимается последовательность действий, которые могут повторяться нужное количество раз. Действия, которые повторяются в цикле, называются телом цикла (команды cin и изменение sum). Применение циклов сокращает длину программы, однако не сокращают время выполнения программы. Считается, что циклы – это самые продолжительные по времени фрагменты программы. Существует несколько разновидностей циклов, все их можно разделить на две группы:

- циклы с условием (while, do).

- цикл с параметром (for);

2. Цикл с предусловием

while (условие цикла)

{

Тело цикла

}

Следующее действие;

Читается: «Пока условие цикла истинно, выполнять команды тела цикла».

Работает так:

1. Сначала вычисляется условие цикла.
2. Если оно истинно (не равно 0), то выполняется тело цикла и переход к 1.
3. Если оно ложно (равно 0), то тело цикла не выполняется и осуществляется выход из цикла на следующее действие. Правила

1. Выражение цикла должно содержать некоторую величину – переменную цикла – значение которой вычисляется в выражении.

2. Тело цикла должно содержать команду, изменяющей значение переменной цикла так, чтобы выражение цикла когда-нибудь стало ложным. Если такой команды не будет, выражение цикла всегда будет истинным и возникнет зацикливание программы - бесконечное повторение тела цикла.

3. Для переменной цикла должны выполниться три действия:

- инициализация переменной цикла – присваивание переменной начального значения – выполняется один раз до начала выполнения тела цикла;
- проверка истинности выражения, в котором участвует переменная цикла – выполняется перед каждым возможным выполнением тела цикла;
- коррекция переменной цикла – изменение значения переменной – выполняется в теле цикла.

Отсутствие хотя бы одного из этих действий может привести к зацикливанию программы.

4. Цикл while является циклом с предусловием, т.к. условие проверяется перед выполнением тела цикла. Значит, тело цикла может ни разу не выполниться, если условие цикла сразу станет ложным.

3. Арифметический цикл

Формат оператора цикла с параметром:

for (выражение_1; выражение_2; выражение_3) оператор;

Цикл с параметром используется в программах, когда известно количество совершаемых повторений. Выражение_1 выполняется только один раз в начале цикла. Обычно оно определяет начальное значение параметра цикла (инициализирует параметр цикла). Выражение_2 — это условие выполнения цикла. Выражение_3 обычно определяет изменение параметра цикла. Оператор — тело цикла, которое может быть простым или составным. В последнем случае используются фигурные скобки.

После вычисления выражения_3 происходит возврат к вычислению выражения_2 — проверке условия повторения цикла.

Используя операцию «запятая», можно в выражение_1 внести инициализацию значений сразу нескольких переменных.

Некоторых элементов в операторе for может не быть, однако разделяющие их точки с запятой обязательно должны присутствовать.

Пример программы с различными циклами

Пример№1. Выдать самый высокий курс $ за 12 месяцев года и указать номер месяца, в котором был получен этот курс. Написать программный код с применением итерационного цикла.

main()

```
{        float kurs, max;
int    i,  nom;  cout<<  "Введите
курс 1 месяца";
cin>> kurs;
max=kurs; nom=1;
i=2;
while (i<=12)
{
cout<< "Введите курс"<< i<<" месяца";
cin>> kurs;  if ( kurs>max)
{ max=ku
rs; nom=i;
} i++;
}
cout<< "самый высокий курс $="<< max;
cout<<”был получен в месяце ”<<nom;
```
} Пример №2. Решить задачу примера №1 с применением арифметического цикла.
```
main()
{
float  kurs,  max;  int    i,  nom;
cout<<    "Введите    курс    1
месяца";
cin>>              kurs;
max=kurs;
nom=1;
for (i=2; i<=12; i++))
{ cout<< "Введите курс"<< i<<"
месяца";  cin>> kurs;  if ( kurs>max)
{ max=kurs;
nom=i;
}
}
cout<<   "самый   высокий   курс   $="<<   max;
cout<<”был получен в месяце ”<<nom;
}
```

Задание

Имеются зарплаты работника по месяцам года – случайные числа в диапазоне от 5000 до 20000. Вывести исходные данные на экран и выполнить для них бухгалтерские расчеты.

На оценку «Удовлетворительно»

1. Вычислить общую сумму выплат за год.
2. Определить среднюю зарплату работника.
3. Вывести самую высокую и низкую зарплату, и в каких месяцах он ее получил.

На оценку «Хорошо»

1. Рассчитать ежемесячные пенсионные отчисления, как 2% от зарплаты сотрудника.
2. Вычислить общую сумму выплат за год в пенсионных фонд.

На оценку «Отлично»

1. Вывести номера месяцев, когда его зарплата была ниже прожиточного минимума (вводится пользователем).

2. Вычислить сумму налога за год. Отчисление начинаются с того месяца, когда общая сумма по зарплате станет более 20 000 и составляет 13%.

Контрольные вопросы

1. Когда в программах используется цикл с параметром?
2. Когда в программах используется цикл с условием?

Лабораторная работа №21.
Программирование алгоритмов со структурой вложенных циклов.

Цель: Получение навыков в работе с арифметическими выражениями, циклическими инструкциями.

Темы для предварительной проработки Арифметические операторы С++.

Функции стандартного ввода и вывода.

Оператор ветвления. **Теоретический материал**

1. Вложенные циклы

Циклы позволяют повторять выполнение любого набора операторов. В частности можно повторять много раз выполнение другого цикла. Такие циклы называются вложенными.

Пример 1. Напечатать числа в виде следующей таблицы

3 3 3 3 3
3 3 3 3 3
3 3 3 3 3
3 3 3 3 3

Таблица состоит из четырех строк, в каждой из которых число 3 напечатано 5 раз. Строку из пяти чисел можно напечатать с помощью одного цикла for:

```
for (i = 1; i <= 5; i++)
cout<<”3 “; cout<<endl;
```

Чтобы повторить вывод строчки 4 раза, вставляем этот цикл внутрь другого for

```
(k = 1; k <= 4; k++)
{
for (i = 1; i <= 5; i++)
cout<<”3 “;
cout<<endl;
}
```

Типичная ошибка, когда в качестве счетчиков вложенных циклов (i и k в приведенном примере) используется одна и та же переменная. То есть нельзя в каждом из циклов использовать одну переменную i. Помнить об этом особенно важно, поскольку данная ошибка не обнаруживается на этапе компиляции. Ваша программа запустится, но делать будет вовсе не то, что вы от нее ждете. В приведенном примере (если допустить ошибку, заменив переменную k на i) внешний цикл выполнится всего 1 раз вместо 4-х. Возможна также ситуация, когда такая ошибка приведет к зацикливанию: внешний цикл будет выполняться бесконечно долго – программа зависнет.

2. Задачи перебора

Имеется целый класс задач, решение которых сводится к перебору различных вариантов, среди которых выбирается такой, который удовлетворяет условию задачи.

Пример 2. Целое число *K* является делителем *N*, если остаток от деления *N* на *K* равен 0. Чтобы найти все делители достаточно перебрать все числа от 1 до *N* и проверить, являются ли они делителями. Данный алгоритм можно реализовать с помощью программы:

```
for (K = 1; K <= N/2; K++)
if (N%K==0) cout<<K;
```

На этом простейшем примере ясна общая структура решения задач на перебор вариантов. Для организации перебора используется цикл, каждый шаг выполнения которого соответствует рассмотрению одного из вариантов. Внутри цикла стоит проверка, подходит ли данный вариант под условие задачи.

Такие задачи обычно решается с помощью вложенных циклов, в которых в условии цикла используется перебираемый признак.

Пример 3. Пусть двумя числами (H и V) задано положение коня на шахматной доске. Найдите координаты всех клеток, куда конь может пойти следующим ходом (других фигур на доске нет).

В данном примере, вариантами решения являются координаты клеток, то есть каждый вариант это не одно число, а два. Необходимо придумать способ вычисления варианта (в данном случае номеров горизонтали и вертикали) по его номеру.

Известно, что конь ходит буквой Г, смещаясь на две клетки в одном направлении и на одну в другом. Пусть проверяется клетка с координатами (X, Y). Тогда условие сдвига по вертикали на 2 будет выглядеть так: abs(x-2).

Взятие модуля необходимо, чтобы учесть возможность сдвига как вправо, так и влево. Аналогично условие сдвига на одну клетку, например, по вертикали: abs(y-2).

Полное условие возможности пойти на клетку конем будет выглядеть как:

```
((abs(X-H)==2)&&(abs(Y-V)==1)) || ((abs(Y-V)==2)&&(abs(X-H)==1))
```

Чтобы перебрать все клетки шахматной доски удобно для перебора использовать вложенные циклы. В данном случае номера вертикали и горизонтали являются целым числами от 1 до 8, поэтому для перебора их значений можно использовать счетчики циклов:

```
cin>>H>>V; for (X = 1;
X <= 8; X++) for (Y =
1; Y <= 8; Y++)
if ((abs(X-H)==2)&&(abs(Y-V)==1)) || ((abs(Y-V)==2)&&(abs(X-H)==1))
cout<<X<<Y<<endl;
```

Пример программы с вложенными циклами

Найти все совершенные числа в заданном диапазоне. Совершенным считается число, равное сумме всех своих делителей. Задать диапазон от 1 до 10000.

```
main()
{       int i, j, k=0, sum=0; int
n=10000;
cout<<"\n Diapason 1- 10000";
for(i=1; i<=n; i++)
{sum=1;
for (j=2; j<i; j++)
if (i%j==0)
sum+=j; if(i ==
sum)
{k++;
cout<<"\n"<<i;
}
}
```

```
cout<<"\nkol
sov="<<k; }
```

Результат работы программы:

```
Diapason 1- 10000
1
6
28
496 8128
kol
sov=5
```

Задание

На оценку «Удовлетворительно»

Написать программу разложение число на простые множители.

На оценку «Хорошо»

Написать программу нахождения НОД разложением на простые множители.

На оценку «Отлично»

Составить программу-генератор пифагоровых троек. Пифагоровой тройкой называют такие целые числа a, b и c, которые удовлетворяют условию. Такие числа могут быть сторонами прямоугольного треугольника. Найдем такие числа.

Контрольные вопросы

1. Какие задачи называются задачами перебора? 2. Объяснить, как работает конструкция вложенных циклов.

Лабораторная работа №22.
Реализация задач перевода целых величин в различные системы счисления.

Цель: Получение навыков в работе с системами счисления и итерационными циклами.

Темы для предварительной проработки Арифметические операторы C++.

Функции стандартного ввода и вывода.

Итерационные циклы. **Теоретический материал**

1. Системы счисления

В позиционных системах счисления один и тот же числовой знак (цифра) в записи числа имеет различные значения в зависимости от того места (разряда), где он расположен. К числу таких систем относится современная десятичная система счисления, возникновение которой связано со счётом на пальцах.

Под позиционной системой счисления обычно понимается b-ричная система счисления, которая определяется целым числом b>1, называемым основанием системы счисления. Целое число без знака x в b-ричной системе счисления представляется в виде конечной линейной комбинации степеней числа b:

$$x = \sum_{k=0}^{n-1} a_k b^k$$

a_k — это целые числа, называемые цифрами, удовлетворяющие неравенству $0 \leq a_k \leq (b-1)$

Каждая степень b^k в такой записи называется весовым коэффициентом разряда. Старшинство разрядов и соответствующих им цифр определяется значением показателя k (номером разряда). Обычно в записи ненулевых чисел начальные нули опускаются.

В любой позиционной системе число может быть представлено в виде многочлена. Покажем, как представляют в виде многочлена десятичное число:

$$4567 = 4000 + 500 + 60 + 7 = 4 \cdot 10^3 + 5 \cdot 10^2 + 6 \cdot 10^1 + 7 \cdot 10^0$$

Пример программы с системами счисления

Перевести число из системы счисления q в 10-ную систему счисления. main()

```
{       long int x, y, step;
int a, q;
cout<<"Введите систему счисления->";          cin>>q;  cout<<"Введите
число в "<<q<<" системе счисленмя->";  cin>>x;
y=0;                                  //Результат
step=0;                  //Степень основания
while(x!=0)              //Пока в числе есть цифры
{a=x%10;                 //Последняя цифра
if (a>=q)                //Проверка правильности ввода
        {
        cout<<"Error";
        exit(0);
        }
y=y+a*pow (q, step); //Результат – накапливаемая сумма
x=x/10;                  //Убрать последнюю цифру
step++;          //Повысить степень основания
} cout<<"Результат="<<y<<" в 10
c/c"<<endl; }
```

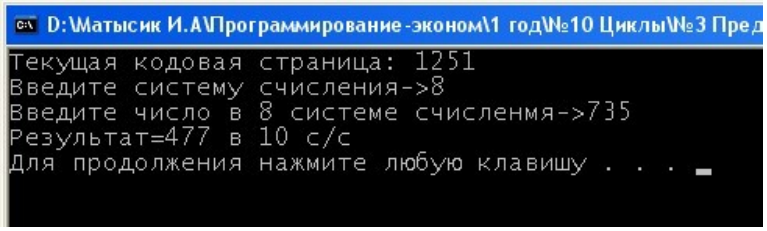


Рисунок 4 – Окно с результатами программы перевода чисел

Задание

На оценку «Удовлетворительно»

Вводится система счисления q и число x в этой системе счисления. Перевести число x в 10тичную систему счисления. Использовать при разработке программы встроенный отладчик. *На оценку «Хорошо»*

Вводится число x в 10 системе счисления. Выбрать нужную систему счисления q и перевести число x в эту систему счисления. Тестировать результаты работы программы ручным подсчетом.

На оценку «Отлично»

Вводится q1 – исходная система счисления и q2 – желаемая система счисления, а так же число x в системе счисления q1. Перевести число x в систему счисления q2.

Контрольные вопросы

1. Как надо задать условие цикла, чтобы перевести число в нужную систему счисления? 2. Какие операции выделяют из цисла его цифры?

Лабораторная работа №23.
Программа с использованием функций без параметров.

Цель: Получение навыков в работе с функциями.

Темы для предварительной проработки

Структура программы на языке С++.

Использование стандартных функций.

Теоретический материал

1. Понятие и назначение функций

Любая С++-программа составляется из "строительных блоков", именуемых функциями. Функция — это подпрограмма, которая содержит одну или несколько С++-инструкций и выполняет одну задачу.

Каждая функция имеет имя, которое используется для ее вызова. Своим функциям программист может давать любые имена за исключением имени main(), зарезервированного для функции, с которой начинается выполнение программы.

Функции — это "строительные блоки" С++-программы.

В С++ ни одна функция не может быть встроена в другую. Одна функция может вызывать другую.

В уже рассмотренных примерах программ функция main() была единственной, функция main () — первая функция, выполняемая при запуске программы. Ее должна содержать каждая С++-программа. Функции бывают двух типов. К первому типу относятся функции, написанные программистом (main() — пример функции такого типа). Функции другого типа находятся в стандартной библиотеке С++-компилятора -. коллекция встроенных функций. Как правило, С++программы содержат как функции, написанные программистом, так и функции, предоставляемые компилятором.

Эта программа содержит две функции: main() и myfunc().

```
void myfunc();          // прототип функции myfunc()
main() { cout << "В функции main() . " ;
myfunc();               // Вызываем функцию myfunc() •
cout << "Снова в функции main () ";
}
void myfunc()
{ cout << " В функции myfunc() .
"; }
```

Программа работает следующим образом. Вызывается функция main() и выполняется ее первая cout-инструкция. Затем из функции main() вызывается функция myfunc (). Для вызова функции в программе указывается имя функции myfunc, за которым следуют пара круглых скобок и точка с запятой. Затем функция myfunc() выполняет свою единственную cout-инструкцию и передает управление назад функции main(), причем той строке кода, которая расположена непосредственно за вызовом функции. Наконец, функция main() выполняет свою вторую coutинструкцию, которая завершает всю программу. Результаты работы программы:

В функции main() . В функции myfunc() . Снова в функции main().

2. Прототип функции

Прототип функции объявляет функцию до ее определения и использования. Прототип позволяет компилятору узнать тип значения, возвращаемого этой функцией, а также количество и тип параметров, которые она может иметь. Компилятору нужно знать эту информацию до первого вызова функции. Поэтому прототип располагается до функции main (). Единственной функцией, которая не требует прототипа, является main(), поскольку она встроена в язык С++.

```
void myfunc();          // прототип функции myfunc()
```

Функция myfunc () не содержит инструкцию return. Ключевое слово void, которое предваряет как прототип, так и определение функции myfunc(),заявляет о том, что функция myfunc

() не возвращает никакого значения. В C++ функции, не возвращающие значений, объявляются с использованием ключевого слова void.

Пример программы с функциями без параметров

Составить программу с функцией, которая рисует на экране строку, состоящую из 80 звездочек.

```
void line(void);                    //Прототип функции line main()
                                    //Основная функция
{
line();                             //Вызов функции line
}
//Определение функции line
void line(void)
{       int   i;
for (i=0;   i<80;   i + +)
cout<<"*";
}
```

Задание

На оценку «Удовлетворительно»

Составить программу с тремя функциями, выводящими на экран строки символов #, @, $.

На оценку «Хорошо»

Составить программу для вычисления площади кольца по значениям внутреннего и внешнего радиусов, используя функцию вычисления площади круга.

На оценку «Отлично»

Даны три целых числа. Определить, сумма цифр которого из них больше. Подсчет суммы цифр организовать через функцию.

Контрольные вопросы

1. Укажите назначений функций в программе.
2. Сколько минимально функций может быть в программе?
3. Зачем нужен прототип функции?

Лабораторная работа №24.
Программа с использованием функций с параметрами.

Цель: Получение навыков в работе с функциями.

Темы для предварительной проработки Структура программы на языке С/С++.

Функции без параметров.

Теоретический материал

1. Назначение аргументов функции

Функции можно передать одно или несколько значений. Значение, передаваемое функции, называется аргументом. **Аргумент** — это значение, передаваемое функции при вызове.

Рассмотрим программу, которая для отображения абсолютного значения числа использует стандартную библиотечную функцию abs(). Эта функция принимает один аргумент, преобразует его в абсолютное значение и возвращает результат.

```
main()
{
```

```
cout << abs(-
10); }
```

Здесь функции abs() в качестве аргумента передается число -10. Функция abs() принимает этот аргумент при вызове и возвращает его абсолютное значение, которое в данном случае равно числу 10. Если функция принимает аргумент, он указывается внутри круглых скобок, расположенных сразу после имени функции.

Рассматриваемая программа включает заголовок <stdlib.h>. Он необходим для обеспечения возможности вызова функции abs(). Каждый раз, когда используется библиотечная функция, в программу необходимо включать соответствующий заголовок, который помимо прочей информации, содержит прототип библиотечной функции.

2. Передаваемые параметры

Параметр — это определяемая функцией переменная, которая принимает передаваемый функции аргумент.

При создании функции, которая принимает один или несколько аргументов, иногда необходимо объявить переменные, которые будут хранить значения аргументов. Эти переменные называются параметрами функции. Например, следующая функция выводит произведение двух целочисленных аргументов, передаваемых функции при ее вызове.

```
void mul(int x, int y)
{ cout << x * y << "
"; }
```

При каждом вызове функции mul() выполняется умножение значения, переданного параметру x, на значение, переданное параметру y. Рассмотрим программу, которая демонстрирует использование функции mul().

```
void mul(int x, int y);          // Прототип функции mul() .
main() {

mul(10, 2                  //Вызовы функции с параметрами
0); mul(5,
6); mul (8,
9); }
```

Эта программа выведет на экран числа 200, 30 и 72. При вызове функции mul() C++ +компилятор копирует значение каждого аргумента в соответствующий параметр. В данном случае при первом вызове функции mul() число 10 копируется в переменную x, а число 20 — в переменную y. При втором вызове 5 копируется в x, а 6 — в y. При третьем вызове 8 копируется в x, а 9 — в y.

3. Функции, возвращающие значения

В C++ для возврата значения из функции используется инструкция return. Общий формат этой инструкции таков:

return значение;

Значение представляет собой значение-результат, возвращаемое функцией.

Исправим функцию mul() так, чтобы она возвращала результат - произведение своих аргументов.

```
int mul(int x, int y);          // Прототип функции mul() с результатом main()
{       int answer;
answer = mul(10, 11);          //Вызов с возвратом значения
cout << "Ответ равен " <<  answer; }
// Эта функция возвращает значение.
int mul(int x, int y)
{
return x * y;                  // Функция возвращает произведение x и y.
```

```
}
```

В этом примере функция mul() возвращает результат вычисления выражения x*y с помощью инструкции return. Затем значение этого результата присваивается переменной answer.

Поскольку в этой версии программы функция mul () возвращает значение, ее имя в определении не предваряется словом void. Здесь функция mul() возвращает значение целочисленного типа. Тип значения, возвращаемого функцией, предшествует ее имени, как в прототипе, так и в определении.

При достижении инструкции return функция немедленно завершается, а весь остальной код игнорируется. Функция может содержать несколько инструкций return. Возврат из функции можно обеспечить с помощью инструкции return без указания возвращаемого значения, но такую ее форму допустимо применять только для функций, которые не возвращают никаких значений и объявлены с использованием ключевого слова void.

Пример программы с функциями с параметрами

Составить программу нахождения наибольшего значения из трех величин — max (a, b, c). Для ее решения можно использовать вспомогательный алгоритм нахождения максимального значения из двух, поскольку справедливо равенство: max (a, b, c) = max (max (a, b), c).

```
//Определение вспомогательной функции
int MAX(int x, int y)
{ if
(x>y)
return x;
else
return y;
}
//Основная функция
main()
{ int a,b,c,d;
cout<<"Введите a,b,c:";
cin>>a>>b>>c; d=MAX
(MAX (a, b), c); cout<<"\
nmax (a, b, c) ="<<d;
}
```

Задание

На оценку «Удовлетворительно»

Составить программу с функцией, выводящей на экран строку нужной длины нужным символом. Функция принимает два параметры – длину строки и символ. Возвращаемых значений нет.

На оценку «Хорошо»

Составить программу для вычисления площади кольца по значениям внутреннего и внешнего радиусов, используя функцию вычисления площади круга. Функции передается один параметр – радиус и возвращается результат – вычисленная площадь. *На оценку «Отлично»*

Даны три целых числа. Определить, сумма цифр которого из них больше. Подсчет суммы цифр организовать через функцию, которая имеет два входных параметра – складываемые числа и результат – вычисленную сумму.

Контрольные вопросы

1. Что такое аргумент функции и зачем он используется?
2. Что такое параметр функции и как им пользоваться?
3. Как из функции вернут результат ее работы?

Лабораторная работа №25. Рекурсивные функции.

Цель: Получение навыков в работе с рекурсивными функциями.

Темы для предварительной проработки Основные конструкции С++.

Функции.

Циклические алгоритмы.

Теоретический материал

1. Понятие рекурсивной функции

Рекурсией называется ситуация, когда подпрограмма вызывает сама себя. Функция может содержать вызов других функций. В том числе процедура может вызвать саму себя. Пример рекурсивной функции:

```
int Rec(int a)
{
if (a>0) Rec(a-
1); cout<<a; }
```

Функция Rec вызывается с параметром a = 3. В ней содержится вызов функции Rec с параметром a = 2. Предыдущий вызов еще не завершился, поэтому создается еще одна функция и до окончания ее работы первая свою работу не заканчивает. Процесс вызова заканчивается, когда параметр a = 0. В этот момент одновременно выполняются 4 экземпляра функции. Количество одновременно выполняемых функций называют **глубиной рекурсии**. Четвертая вызванная процедура (Rec(0)) напечатает число 0 и закончит свою работу. После этого управление возвращается к функции, которая ее вызвала (Rec(1)) и печатается число 1. И так далее пока не завершатся все вызовы. Результатом исходного вызова будет печать четырех чисел: 0, 1, 2, 3.

Пример программы с рекурсивной функцией

Если функция вызывает сама себя, то, по сути, это приводит к повторному выполнению содержащихся в ней инструкций, что аналогично работе цикла.

Для примера сымитируем работу цикла for. Для этого потребуется переменная счетчик шагов, которую можно реализовать как параметр функции.

```
//имитация работы цикла с помощью рекурсии
void   LoopImit(int i, int n)
{
cout<<"\n"<<"  1 n "<<" i "<<i;
//Здесь может располагаться первый блок инструкций
if (i<=n)
LoopImit(i+1, n);
cout<<"\n"<<"  2 n "<<" i "<<i;
//Здесь может располагаться второй блок инструкций
}
main()
{         int i=0, n;
cout<<"n=";
cin>>n;
LoopImit(i, n); }
```

Результат работы программы:

```
n=5
1 n  i 0
```

1 n i 1
1 n i 2
1 n i 3
1 n i 4
1 n i 5
1 n i 6
2 n i 6
2 n i 5
2 n i 4
2 n i 3
2 n i 2
2 n i 1
2 n i 0

Задание

На оценку «Удовлетворительно»

Разработать рекурсивную функцию вычисления факториала.

На оценку «Хорошо»

Разработать рекурсивную функцию, которая для двух заданных натуральных чисел определяет их наибольший общий делитель.

На оценку «Отлично»

Разработать рекурсивную функцию, определяющую число Фибоначчи с номером n.

Контрольные вопросы

1. Что такое рекурсивная функция?
2. Что может заменить рекурсия?

Лабораторная работа №26. Рекурсия. Ханойские башни.

Цель: Получение навыков в работе с рекурсивными функциями.

Темы для предварительной проработки Основные конструкции C++.

Функции.

Теоретический материал

1. Понятие рекурсивной функции

Рекурсией называется ситуация, когда подпрограмма вызывает сама себя. Функция может содержать вызов других функций. Пример рекурсивной функции:

```
int Rec(int a)
{
if (a>0) Rec(a-
1); cout<<a; }
```

Рассмотрим принцип работы рекурсивной функции на примере вызова созданной функции Rec(3).

Функция Rec вызывается с параметром a = 3. В ней содержится вызов функции Rec с параметром a = 2. Предыдущий вызов еще не завершился, поэтому создается еще одна функция и до окончания ее работы первая свою работу не заканчивает. Процесс вызова заканчивается, когда параметр a = 0. В этот момент одновременно выполняются 4 экземпляра функции. Количество одновременно выполняемых процедур называют **глубиной рекурсии**. Четвертая вызванная функция (Rec(0)) напечатает число 0 и закончит свою работу. После этого управление возвращается к функции, которая ее вызвала (Rec(1)) и печатается число 1. И так далее пока не

завершатся все вызовы. Результатом исходного вызова будет печать четырех чисел: 0, 1, 2, 3. *2. Задача о Ханойских башнях*

Существует древнеиндийская легенда, согласно которой в городе Бенаресе под куполом главного храма, в том месте, где находится центр Земли, на бронзовой площадке стоят три алмазных стержня. В день сотворения мира на один из этих стержней было надето 64 кольца. Бог поручил жрецам перенести кольца с одного стержня на другой, используя третий в качестве вспомогательного. Жрецы обязаны соблюдать условия:

1. переносить за один раз только одно кольцо;
2. кольцо можно переносить с одного стержня (x) на другой (y), только если оно имеет меньший диаметр, чем верхнее кольцо, находящееся на стрежне y, или стержень y свободен.

Согласно легенде, когда, соблюдая все условия, жрецы перенесут все 64 кольца, наступит конец света. Минимальное число ходов, необходимое для решения головоломки, равно $2^n - 1$, где n — число дисков. Если считать, что на перенос одного кольца потребуется одна секунда, то на перенос всех 64 колец потребуется более 5 млрд веков.

Если башня состоит из одного диска, то она переносится за один ход: 1->3.

Башня из двух дисков переносится за три хода: 1—>2, 1—>3, 2—>3.

Для переноса башни из трех дисков потребуется уже семь ходов: 1->3, 1->2, 3->2, 1->3, 2->1, 2->3, 1->3.

Чтобы перенести башню из четырех дисков с первого стержня на третий, необходимо действовать по плану:

1) перенести башню из трех верхних дисков с первого стержня на второй (7 ходов);

2) самый большой диск перенести с первого стержня на третий (1 ход); 3)

перенести башню из трех дисков со второго стержня на третий (7 ходов).

Всего на перенос потребуется 15 ходов.

Рассуждая аналогичным образом, сосчитаем число ходов, необходимых для переноса башни из пяти дисков: 15 + 1 + 15 = 2 • 15 + 1 = 31.

Рассмотренный алгоритм решения задачи «Ханойская башня» обладает рекурсивным свойством: в ходе его выполнения для башни, состоящей из n колец, используется алгоритм для чуть более простой ситуации — переноса башни, состоящей из n - 1 кольца. В свою очередь, в алгоритме для башни из n - 1 кольца используется этот же алгоритм для n - 2 колец и т. д.

Пример программы с рекурсивной функцией

```
//ханойские башни
void hb(int n,int x,int y,int z)
{       if
(n>0)
{hb (n-1, x, z, y);
cout<<"\n<P> "<< x<< " -> "<<y;
hb (n-1, z, y, x);
} }
main(
)
{       int st1, st2, st3, st4;
//Введите число колец st1
cout<<"\nKol rkolez ";
cin>>st1;
//Введите начальный стержень st2
cout<<"\nnum beg "; cin>>st2;
```

```
//Введите конечный стержень st3
cout<<"\nnum end "; cin>>st3;
//Введите вспомогательный стержень st4
cout<<"\nnum wrem "; cin>>st4;
hb(st1,st2,st3,st4);
}
```

Результаты работы программы

```
Kol rkolez 3
num beg 1 num
end 3 num
wrem 2
<P> 1 -> 3
<P> 1 -> 2
<P> 3 -> 2
<P> 1 -> 3
<P> 2 -> 1
<P> 2 -> 3
<P> 1 -> 3
```

Задание

На оценку «Удовлетворительно»

Напишите рекурсивную функцию, определяющую количество единиц в двоичном представлении натурального числа.

На оценку «Хорошо»

Напишите рекурсивную функцию, которая по заданным натуральным числам m и n выводит все различные представления числа «в виде суммы m натуральных слагаемых». Представления, различающиеся лишь порядком слагаемых, считаются одинаковыми.

На оценку «Отлично»

Напишите рекурсивную функцию, которая переворачивает элементы массива в обратном порядке.

Контрольные вопросы

1. Что такое рекурсивная функция?
2. Расскажите суть задачи о Ханойских башнях и способе ее решения.

Лабораторная работа №27.
Программа, реализующая основные операции с одномерным массивом.

Цель: Получение навыков в работе с массивами, выполнения над ними базовых операций.

Темы для предварительной проработки Типы данных в языке C/C++.

Функции стандартного ввода и вывода.

Оператор ветвления.

Операторы циклов.

Теоретический материал

1. Понятие одномерного массива

Массив — это структура однотипных элементов, занимающих непрерывную область памяти. С массивом связаны следующие его свойства: имя, тип, размерность, размер.

Формат описания массива следующий:

тип элементов имя [константное_выражение];

Константное выражение определяет размер массива, т. е. число элементов этого массива. Например, согласно описанию int A[10];

объявлен массив с именем A, содержащий 10 элементов целого типа. Элементы массива обозначаются индексированными именами. Нижнее значение индекса равно 0:

A[0], A[1], A[2], A[3], A[4], A[5], A[6], A[7], A[8], A[9].

В C++ нельзя определять произвольные диапазоны для индексов. Размер массива, указанный в описании, всегда на единицу больше максимального значения индекса.

Размер массива может явно не указываться, если при его объявлении производится инициализация значений элементов. Например:

int p[]={2, 4, 6, 10, 1};

В этом случае создается массив из пяти элементов со следующими значениями: p[0]=2, p[1]=4, p[2]=6, p[3]=10, p[4]=1.

В результате следующего объявления массива

int M[6]={5, 3, 2} ;

будет создан массив из шести элементов. Первые три элемента получат инициализированные значения. Значения остальных будут либо неопределенными, либо равны нулю, если массив внешний или статический.

2. Основные действия с одномерным массивом Пример 1. Ввод с клавиатуры и вывод на экран одномерного массива.

```
main()
{       int i, A[5];
for (i=0; i<5; i++)
{
cout<<"A["<<i<<"]=";
cin>>A[i];
}
for (i=0; i<5; i++)
cout<<A [ "<<i<<" ] ="<<A [ i]<<"; }
```

Пример 2. Ввод вещественного массива и вычисление среднего значения.

```
main()
{ const n=10;
int i;
double A[n], SA;
for(i=0; i<n; i + +)
{
cout<<"A [ "<<i<<" ] =" ;
cin>>A[i] ;
}
SA=0;
for(i=0; i<n;i++)
SA=SA+A[i]; SA=SA/n;
cout<<"/среднее
значение="<<SA;
}
```

В этой программе обратите внимание на определение размера массива через константу.

Пример программы с одномерными массивами

Составить программу, подсчитывающую число минимальных элементов в одномерном массиве.

```
main()
```

```
{         const n=10;
int      a[n]={25,3,16,-2,1,10,0,5,-2,10};
 int i,min,k;
for (i=0; i<n; i++) cout<<a[i]<<" ";
min=a[0];              k=1;
for(i=0; i<n; i++)
{ if
(a[i]<min)
{min=a[i];
k=0; }
if  (a[i]==min)  k+
+;
}
cout<<"\n min="<<min<<" kol="<<k;
}
```

Результаты работы программы
25 3 16 -2 1 10 0 5 -2 10 min=-2
kol=2

Задание

На оценку «Удовлетворительно»

В целочисленной последовательности есть нулевые элементы. Создать массив из номеров этих элементов.

На оценку «Хорошо»

Дана последовательность действительных чисел a_1, a_2,..., a_n. Заменить все ее члены, большие данного Z, этим числом. Подсчитать количество замен.

На оценку «Отлично»

Дан массив действительных чисел, размерность которого N. Подсчитать, сколько в нем отрицательных, положительных и нулевых элементов

Контрольные вопросы

1. Что такое одномерный массив?
2. Как описать одномерный массив в программе? 3. Как осуществить доступ к отдельному элементу массива?

Лабораторная работа №28.
Применение одномерных массивов в математической статистике.

Цель: Получение практических навыков работы с одномерными массивами и их прикладное применение в задачах дисциплины «Математическая статистика».

Темы для предварительной проработки Типы данных в языке С/С++.

Функции стандартного ввода и вывода.

Оператор ветвления.

Операторы циклов.

Теоретический материал

1. Понятие одномерного массива

Массив — это структура однотипных элементов, занимающих непрерывную область памяти. С массивом связаны следующие его свойства: имя, тип, размерность, размер.

Формат описания массива следующий:

тип элементов имя [Размер];

Размер массива - число элементов этого массива. Например, согласно описанию

int A[10];

объявлен массив с именем A, содержащий 10 элементов целого типа. Элементы массива обозначаются индексированными именами. Нижнее значение индекса равно 0:

A[0], A[1], A[2], A[3], A[4], A[5], A[6], A[7], A[8], A[9].

В C++ нельзя определять произвольные диапазоны для индексов. Размер массива, указанный в описании, всегда на единицу больше максимального значения индекса.

2. Роль ЭВМ в статических расчетах

Цель математической статистики – описание, объяснение и предсказание явлений действительности на основе установленных знаков, что позволяет находить решения в типичных ситуациях. В основе научных знаний лежит наблюдение.

Математическая статистика изучает методы обработки результатов наблюдений массовых случайных явлений, обладающих статистической устойчивостью, закономерностью, с целью выявления этой закономерности. Выводы о закономерностях, которым подчиняются явления, изучаемые методами математической статистики, всегда основываются на ограниченном, выборочном числе наблюдений.

Для решения многих практических задач совсем необязательно знать все возможные значения случайной величины, а достаточно указать отдельные числовые параметрыхарактеристики, такие как математические ожидание, дисперсия, моменты различных порядков и

т.д. Эти характеристики случайной величины называют числовыми характеристиками случайной величины. Подсчет значений этих характеристик ведется по формулам с фиксированным количеством наблюдений, на основе которых могут быть построены вычислительные алгоритмы.

Огромное значение в настоящее время имеет использование вычислительной техники, как средства автоматизации ручных громоздких подсчетов характеристик случайной величины. Достаточно создать универсальный программный продукт для подсчета необходимых числовых величин и затем пользоваться им необходимое количество раз для обработки различных исходных данных.

3. Числовые характеристики случайных величин

Математическое ожидание

Дисперсия

Среднее квадратичное отклонение

Начальный момент к-порядка

Центральный момент k-го порядка

Коэффициент асимметрии

$$Mx = \sum_{i=1}^{n} x_i p_i$$

$$Dx = \sum_{i=1}^{n} (x_i - Mx)^2 p_i$$

$$\delta_x = \sqrt{Dx}$$

$$\vartheta_k = Mx^k$$

$$\mu_k = \sum_{i=1}^{n} (x_i - Mx)^k p_i$$

$$A = \mu_3/\delta_x^3$$

$$E = \frac{\mu_4}{\delta_x^4} - 3$$

Эксцесс

4. Применение одномерных массивов в статистических расчетах

В дисциплине «Математическая статистика» применяются одномерные массивы для хранения и обработки последовательности чисел. В общем виде, структуры входных данных программы можно представить следующим образом: #define N число_компонент_ в_векторе

Тип_элементов имя[N];

Например, можно записать так:

```
#define N 10
int   x[N];
float p[N];
```

После выделения памяти под массив X и P, необходимо заполнить ее исходными данными. Наиболее универсальным алгоритм будет в случае, когда исходные данные вводит сам пользователь программы путем набора их с клавиатуры. Это обеспечит использование одной и той же программы многократно для различных исходных данных. При вводе исходных данных необходимо внимательно следить за правильностью набора и своевременной корректировке их. Данные набираются в строчку, отделяя числа в одной строке пробелом. В конце набора данных нажимается клавиша [Enter]. На этом заканчивается первый шаг алгоритма решаемой задачи, целью которого являлось ввод исходных данных в программу и сохранение их в массивах.

После ввода исходных данных в одномерный массив, неоходимо их обработать, реализуя вычисления математических характеристик, путем программирования формулы, представленные выше.

Вычисление таких величин, как математическое ожидание, дисперсия, начальные и центральные моменты требуют реализации алгоритма подсчета суммы из конечного числа слагаемых. Алгоритм вычисления суммы реализуется следующими шагами: сумма=0;

```
//Обнуление суммы до цикла for( i=0; i< N; i++)
 {
 Получить слагаемое
сумма=сумма+слагаемое;
}
```

Рассмотрим для примера нахождение математического ожидания. В качестве суммы здесь выступает величина mx, слагаемое определяется формулой x[i]*p[i], где x[i] и p[i] – это компоненты массивов x и p соответственно, стоящие в последовательности под номером i. Перебирая в цикле for значения всех номеров последовательности, можно тем самым обработать всю совокупность чисел.

```
mx=0;
for(i=0; i< N; i++)  mx=mx+x[i]*p[i];
```

Похожие действия проводят для вычисления дисперсии и моментов. Отличают они только видом слагаемого в сумме. Остальные числовые характеристики, такие как средне квадратичное отклонение, коэффициент асимметрии и эксцесс не требует вычисления суммы, поэтому и вычисляются последовательно единственным оператором присвоения.

Пример программы с применением одномерных массивов

Составить программу, выполняющую расчеты статистических характеристик для следующих исходных данных

Xi	1	10	7	10	4	2	5	7	13	14
Pi	0,02	0,08	0,20	0,15	0,05	0,25	0,09	0,01	0,07	0,08

```
#define N 10   //Размерность массивов main()
{       int       x[N];
```

```
float p[N];
float mx, dx, gx, u1, u2, u3, u4, m1, m2, m3, m4, a, e;
int i;
//ввод с клавиатуры исходных данных
cout<<“Введите в строчку<<”<<N<<”выборок”<<endl;
for ( i= 0; i<N; i++)    cin>>x[i]; cout<<“Введите в
строчку<<N<<” вероятностей”<<endl; for ( i= 0; i<N; i+
+)        cin>>p[i];
//вычисление математического ожидания mx=0;
for(i=0; i< N; i++)      mx=mx+x[i]*p[i];
//вычисление дисперсии
dx=0; for(i=0; i< N; i++)       dx=dx+sqr(x[i]-
mx)*p[i]; //вычисление средне квадратичного
отклонения gx=sqrt(dx);
//вычисление начальных моментов 1, 2, 3, и 4-ых порядков u1=mx;
//совпадает со значением мат. ожидания u2=u3=u4=0;
for(i=0; i< N; i++)
{u2=u2+sqr(x[i])*p[i];
u3=u3+sqr(x[i])*x[i])*p[i];
u4=u4+sqr(x[i])*sqr(x[i])*p[i];
}
//вычисление центральных моментов 1, 2, 3 и 4-ых порядков m1=0;
          //всегда равен 0
m2=dx;                    //совпадает со значением дисперсии m3=m4=0;
for (i=0; i< N; i++)
{m3=m3+sqr(x[i]-mx)*(x[i]-mx)*p[i];
m4=m4+sqr(x[i]-mx)*sqr(x[i]-mx)*p[i]; }
//вычисление коэффициента асимметрии
a=m3/(sqr(gx)*gx); //вычисление
аксцесса e=m4/(sqr(gx)*sqr(gx))-3;
//вывод результатов работы программы на монитор
cout<<“Математическое ожидание =”<< mx<<endl;
cout<<“Дисперсия =”<<dx<<endl;
cout<<“Средне квадратичное отклонение =”<<gx<<endl;
cout<<“Нач.момент 1-го порядка=”<< u1<<endl;
cout<<“Нач.момент 2-го порядка=”<< u2<<endl;
cout<<“Нач.момент 3-го порядка=”<< u3<<endl;
cout<<“Нач.момент 4-го порядка=”<< u4<<endl;
cout<<“Цент.момент 1-го порядка=”<<m1<<endl;
cout<<“Цент.момент 2-го порядка=”<<m2<<endl;
cout<<“Цент.момент 3-го порядка=”<<m3<<endl;
cout<<“Цент.момент 4-го порядка=”<< m4<<endl;
cout<<“Коэффециент асимметрии =”<<a<<endl; cout<<“Аксцесс
=”<<e<<endl;
}
```

Получение результатов работы программы Для рассматриваемого примера результаты выглядят следующим образом:

```
Введите  в строчку 10 выборок
11  4  9  1  5  2  5  2  7
Введите  в строчку 10 вероятностей
0.222 0.034 0.166 0.025 0.075 0.014 0.061 0.002 0.023
```

Математическое ожидание = 6.10400000
Дисперсия = 11.61118400
Средне квадратичное отклонение = 3.40751875
Нач. момент 1-го порядка = 6.10400000
Нач. момент 2-го порядка = 48.87000000
Нач. момент 3-го порядка = 453.92000000
Нач. момент 4-го порядка = 4519.25399999
Центр. момент 1-го порядка = 0.00000000
Центр. момент 2-го порядка = 11.61118400
Центр. момент 3-го порядка = 13.86818573
Центр. момент 4-го порядка = 196.71519835
Коэффециент асимметрии = 0.35051361
Аксцесс = -1.54090053

Задание

Разработать программу для вычисления основных числовых характеристик для дтскретной случайной величины с помощью статистических данных. Написать программу, реализующую алгоритм вычисления указанных величин. Сравнить результаты, полученные ручным подсчетом и с помощью вычислительной техники. Проанализировать полученные данные. Сделать вывод о точности расчетов. Для обработки статистических данных использовать одномерные массивы необходимой размерности.

На оценку «Удовлетворительно»

Рассчитать: математическое ожидание, дисперсию и средне квадратичное отклонение.

На оценку «Хорошо»

Расчитать начальные моменты 1-го, 2-го, 3-го и 4-го порядка.

На оценку «Отлично»

Расчитать центральные моменты 1-го, 2-го, 3-го и 4-го порядка, а так же коэффециент асимметрии и аксцесс.

Варианты индивидуальных заданий

Вариант №1.

X	1	10	7	10	4	2	5	7	13	14
P	0,02	0,08	0,20	0,15	0,05	0,25	0,09	0,01	0,07	0,08

Вариант №2.

X	1	2	6	0	9	4	0	5	7	5
P	0,025	0,011	0,089	0,075	0,01	0,09	0,001	0,005	0,094	0,6

Вариант №3.

X	3	11	4	9	1	5	2	5	2	7
P	0,378	0,222	0,034	0,166	0,025	0,075	0,014	0,061	0,002	0,023

Вариант №4.

X	0	11	10	13	6	4	6	15	8	6
P	0,004	0,01	0,145	0,018	0,124	0,432	0,2	0,01	0,007	0,05

Вариант №5.

X	2	6	11	8	10	13	14	2	4	14
P	0,014	0,543	0,331	0,001	0,007	0,004	0,011	0,044	0,003	0,042

Вариант №6.

X	2	5	7	4	3	5	4	3	1	4
P	0,1	0,017	0,322	0,23	0,012	0,042	0,06	0,023	0,04	0,154

Вариант №7.

X	4	5	1	4	7	2	8	6	6	4
P	0,003	0,176	0,271	0,001	0,41	0,086	0,008	0,04	0,002	0,003

Вариант №8.

X	12	8	12	11	15	5	11	3	0	4
P	0,047	0,064	0,128	0,15	0,512	0,04	0,017	0,02	0,003	0,019

Вариант №9.

X	2	13	1	9	15	4	6	5	9	3
P	0,063	0,014	0,056	0,133	0,266	0,232	0,037	0,02	0,079	0,1

Вариант №10.

X	1	4	6	1	9	8	6	6	1	4
P	0,238	0,418	0,012	0,014	0,004	0,001	0,13	0,11	0,003	0,07

Вариант №11.

X	6	2	5	13	6	12	5	10	4	11
P	0,05	0,025	0,115	0,103	0,169	0,099	0,404	0,005	0,024	0,006

Вариант №12.

X	5	2	2	13	9	10	0	13	4	6
P	0,137	0,34	0,003	0,008	0,231	0,085	0,014	0,176	0,005	0,001

Вариант №13.

X	10	6	5	1	1	4	10	1	3	4
P	0,043	0,076	0,196	0,034	0,072	0,435	0,102	0,004	0,034	0,004

Вариант №14.

X	1	9	15	1	13	6	11	11	5	6
P	0,033	0,238	0,104	0,05	0,176	0,025	0,11	0,232	0,002	0,03

Вариант №15.

X	8	4	1	3	6	9	15	3	11	13
P	0,04	0,085	0,176	0,063	0,15	0,238	0,006	0,137	0,07	0,035

Контрольные вопросы

1. Что такое одномерный массив?
2. Как описать одномерный массив в программе?
3. Для каких вычислений в дисциплине «Математическая статистика» может применяться одномерный массив?

Лабораторная работа №29.
Применение одномерных массивов для моделирования операций с множествами.

Цель: Получение навыков в работе с числовыми одномерными массивами и моделирование на их основе основных операций с множествами.

Темы для предварительной проработки Одномерный массив.

Функции.

Теоретический материал

1. Понятие множества

Множество - это совокупность, набор элементов, объединенных общими свойствами. Эти объекты называются элементами множества. Элементы в множество входят по одному разу, т.е. без повторений. Элементы множества можно перечислять в произвольном порядке. Например, если множество – дни недели, то понедельник элемент этого множества.

Надмножествами можно выполнять основные операции теории множеств. Они представлены в таблице.

2. Представление множества в языке С/С++

В языке программирования нет понятия множества. Однако множество можно задавать в виде одномерных целочисленных массивов, элементы в которых не повторяются. Например, множество 1 1 5 8 12 15 множество 2 8 13 17.

Поэтому основные операции над множествами выполняются над элементами одномерных массивов. Например, для исходных множеств

Результат пересечения множеств 8

Результат объединения множеств 1 5 8 12 15 13 17

Результат разности множеств 1 и 2 1 5 12 15

Пример программы с множествами

Составим программу, демонстрирующую работу создания и объединений множеств.

```
//функция заполнения множества
//Параметры: множество-массив и его размер
void sozd (int a[], int kol) {
int i=0, j, x, pr;
a[i]=random(20);
i++; while(i<kol)
{ x=random(20);
pr=0; /Поиск среди имеющихся элементов в массиве
for(j=0; j<i; j++) if(a[j]==x)
{pr=1; //Такой уже есть
break;
}
if (pr==0) //Такого еще нет
{a[i]=x; //Добавляем его в массив
i++;} }
}
//функция вывода на экран множества
void pr(int a[], int kol)
{ cout<<"\n*** "; for
(int i=0; i<kol; i++)
cout<<a[i]<<" ";
cout<<endl; }
//объединение множеств
void objed(int m1[], int n1,int m2[], int n2, int m3[], int &n3)
{ int i, j, pr;
for (i=0; i<n1; i++) //Копируем первое множество
m3[i]=m1[i]; n3=n1;
for (i=0; i<n2; i++)
{ pr=0
;
```

```
for(j=0; j<n3; j++)              //Поиск среди второго множества
if(m2[i]==m3[j])
{pr=1; break;} if
(pr==0)
{ m3[n3]=m2[i];
n3++;
}
}
}
//Главная функция
main()
{
const n=256;
int       m1[n], m2[n], m3[n], m4[n],m5[n];
int       n1,n2,n3,n4,n5; randomize();
cout<<"\nkol el mas1 "; cin>>n1;
sozd(m1,n1);                     //Вызов – создать первое множество
pr(m1,n1);                              //Вызов – вывод первого множества
cout<<"\nkol el mas2 "; cin>>n2; sozd(m2,n2); pr(m2,n2);
cout<<"\nobjed";
objed(m1,n1,m2,n2,m3,n3); //Вызов – объединение двух множеств
pr(m3,n3);                              //Вызов – вывод результата объединения
}
```

Результат работы программы

```
kol el mas1 7
***   7  14  11  19  1  16  8
kol el mas2 9
***   5  2  8  18  19  10  3  7  4
objed
***  7 14 11 19 1 16 8 5 2 18 10 3 4
```

Задание

На оценку «Удовлетворительно»

Составим программу, демонстрирующую пересечение множеств.

На оценку «Хорошо»

Составим программу, демонстрирующую разность множеств.

На оценку «Отлично»

Составим программу, демонстрирующую симметрическую разность множеств.

Контрольные вопросы

1. Почему можно представить множества одномерными массивами? 2. Какие требования должны соблюдаться для элементов множества при его создании?

Лабораторная работа №30.
Реализация алгоритма пузырьковой сортировки.

Цель: Получение навыков в работе с сортировкой массивов.

Темы для предварительной проработки Типы данных в языке С/С++.

Операторы циклов.

Создание массивов в программе.

Базовые операции с массивами.

Теоретический материал

Алгоритм пузырьковой сортировки

Алгоритм сортировки пузырьковым методом получил свое название от способа, используемого для упорядочивания элементов массива. Здесь выполняются повторяющиеся операции сравнения и при необходимости меняются местами смежные элементы. При этом элементы с меньшими значениями постепенно перемещаются к одному концу массива, а элементы с большими значениями — к другому. Этот процесс напоминает поведение пузырьков воздуха в резервуаре с водой. Пузырьковая сортировка выполняется путем нескольких проходов по массиву, во время которых при необходимости осуществляется перестановка элементов, оказавшихся "не на своем месте". Количество проходов, гарантирующих получение отсортированного массива, равно количеству элементов в массиве, уменьшенному на единицу.

Хотя алгоритм пузырьковой сортировки пригоден для небольших массивов, для массивов большого размера он становится неэффективным. Более универсальным считается алгоритм Quicksort. В стандартную библиотеку C++ включена функция qsort(), которая реализует одну из версий этого алгоритма.

Пример программы, реализующей алгоритм сортировки пузырьком В следующей программе реализована сортировка массива (целочисленного типа), содержащего случайные числа. Эта программа заслуживает внимательного разбора.

```
#define SZ 10
main ()
{       int a[SZ]={30,53,11,35,17,42,21,84,75,61};
int i, j=0;                 //счетчик циклов int k;
//текущий индекс элемента массива int buf;
cout<<"\nСортирозка массива методом \"пузырька\"\n";
for(i=0; i<SZ; i++) cout<<a[i]<<" "; for (i = 0; i < SZ-1;
i++)
{   j+
+;
for (k =0; k < SZ-1; k++)
{
if (a[k] < a[k+1])
{
//обменяем к-й и (к+1)-й элементы
buf = a[k]; a [k] = a[k+1] ; a[k+1] = buf;
}
}
//отладочная печать массива после очередного цикла сортировки
cout<<"\n-----"<<j<<"-----"; for (k = 0; k < SZ; k++) cout<<a[k]<<"
"; cout<<"\n";
}
cout<<"*****Массив отсортирован\n";
for (k =0; k < SZ; k++)
cout<<" "<<a[k];
cout<<"\n\nДля завершения работы нажмите <Enter>";
}
```

Результаты работы программы:

```
Сортировка массива методом "пузырька"
30 53 11 35 17 42 21 84 75 61
```

-----1-----53 30 35 17 42 21 84 75 61 11
-----2-----53 35 30 42 21 84 75 61 17 11
-----3-----53 35 42 30 84 75 61 21 17 11
-----4-----53 42 35 84 75 61 30 21 17 11
-----5-----53 42 84 75 61 35 30 21 17 11
-----6-----53 84 75 61 42 35 30 21 17 11
-----7-----84 75 61 53 42 35 30 21 17 11
-----8-----84 75 61 53 42 35 30 21 17 11 -----9-----
84 75 61 53 42 35 30 21 17 11
*****Массив отсортирован
84 75 61 53 42 35 30 21 17 11

Задание

На оценку «Удовлетворительно»

Сменить порядок сортировки – сделать упорядочение массива по возрастанию значений элементов. Поработать вместо компьютера, проходя инструкции программы. Выполнить отладочную печать.

На оценку «Хорошо»

Изменить принцип заполнения массива – сделать заполнение случайными числами. Проверить результат работы программы.

На оценку «Отлично»

Изменить число элементов массива (100, 1000, 10000, 100000). Исследовать, как изменится время работы программы от размерности массива.

Контрольные вопросы

1. Что такое сортировка массива?
2. Рассказать, в чем суть алгоритма сортировки пузырьком.
3. Указать причины, которые влияют на скорость сортировки массива. 4. Какая существует зависимость между скоростью сортировки и размерностью массива?

Лабораторная работа №31.
Программа, реализующая бинарный поиск.

Цель: Получение навыков в работе с массивами, освоение различных алгоритмов поиска информации в массиве. **Темы для предварительной проработки** Массивы в С++.

Заполнение массива, вывод на экран.

Теоретический материал

1. Понятие и задание массивов в программе

Массив — это структура однотипных элементов, занимающих непрерывную область памяти. С массивом связаны следующие его свойства: имя, тип, размерность, размер.

Формат описания массива следующий:

тип элементов имя [константное_выражение]

Константное выражение определяет размер массива, т. е. число элементов этого массива. Например, согласно описанию int A[10];

объявлен массив с именем А, содержащий 10 элементов целого типа. Элементы массива обозначаются индексированными именами. Нижнее значение индекса равно 0:

A[0], A[1], A[2], A[3], A[4], A[5], A[6], A[7], A[8], A[9]

2. Поиск в массиве методом перебора элементов

Самый простой тип поиска в массиве связан с перебором по порядку всех элементов массива с искомым значение. Такой алгоритм имеет очень простую реализацию, однако, считается одним из самых медленных. Для массивов небольшой размерности он вполне пригоден, однако, для массивов большой размерности его использование не желательно.

```
#define      HB      5
main()
{        int m[HB];              // массив целых int
obr;             // образец для поиска
int found;                       // признак совпадения с образцом int
i ;
cout<<"\nПоиск в массиве методом перебора\n";
cout<<"Введите в одной строке %i целых\
n",HB; cout<<"чисел и нажмите <Enter>\n";
cout<<"->"; for (i = 0; i < HB; i++) cin>>m[i];
cout<<"Введите образец для поиска (целое число)->";
cin>>obr; found = 0;
i = 0;                        // проверяем с первого элемента массива
do{
if (m[i] == obr )
found = 1;                  // совпадение с образцом
else
i++;                          // переход к следующему элементу
} while (!found && i < HB) ;
if ( found )
cout<<"Совпадение с элементом номер %i", i+1; else
cout<<"Совпадений с образцом нет";
}
```

Пример программы, реализующей бинарный поиск в массиве

```
main()
{        const n=1000;
int m[n];
int i, c, k, zc, a, b, kol, pr;
for (i=0;i<n;i++)                  m[i]=i;
cout<<"\n     vvod     0-999     ->";
cin>>k;
 zc=0; kol=0; a=0;      b=n-1; pr=0;
c=(a+b)/2;
cout<<"\n******1/2 diapasona= "<<c<<"\n"; while
(c>=a && c<=b && kol<=n)
{if (m[c]==k)
{ zc=c;
pr=1; break; }
if (k>m[c])
{a=c+1;          c=(a+b)/2;          kol++;
cout<<c<<" ";
}
if (k<m[c])
{b=c-1;          c=(a+b)/2;
```

```
kol++;
cout<<c<<" ";
}
}
if (pr) cout<<"\n poz="<<zc;
else              cout<<"\n no number";
cout<<"\n
kol="<<kol; }
```

Результаты работы программы

1 проверка vvod 0-999 -
>500 ******1/2
diapasona= 499
749 624 561 530 514 506 502 500
poz=500 kol=8
2 проверка vvod 0-999 -
>-100 ******1/2
diapasona= 499
249 124 61 30 14 6 2 0 0
no number kol=9

Задание

На оценку «Удовлетворительно»

Составить программу, реализующую алгоритм линейного поиска. *На оценку «Хорошо»*

Составить программу, реализующую алгоритм бинарного поиска. Выполнить анализ линейного и бинарного поиска по скорости работы с зависимости от размерности массива. *На оценку «Отлично»*

Составить программу, реализующую игру пользователя с компьютером. Компьютер формирует случайное число в диапазоне от 1 до 100 и предлагает пользователю ввести число. Сравнивает свое число и число, введенное пользователем. Выдает итог сравнения – больше, меньше, угадано, подсчитывает число ходов. Идея программы состоит в проверке пользователя использовать метод бинарного поиска.

Контрольные вопросы

1. Рассказать суть метода линейного поиска.
2. Рассказать суть метода бинарного поиска.

Основная литература:

1. Алдан, А. Введение в генерацию программного кода / А. Алдан. - 2-е изд., испр. - М. : Национальный Открытый Университет «ИНТУИТ», 2016. - 189 с. : схем. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=428825

2. Соколов В.П. Кодирование в системах защиты информации [Электронный ресурс]: учебное пособие/ Соколов В.П., Тарасова Н.П.— Электрон. текстовые данные.— М.: Московский технический университет связи и информатики, 2016.— 94 с.— Режим доступа: http://www.iprbookshop.ru/61485.html.— ЭБС «IPRbooks»

3. Антимиров, В. М. Проектирование аппаратуры систем автоматического управления. В 2 ч. Ч. 1 : учебное пособие для СПО / В. М. Антимиров. — 2-е изд. — Саратов, Екатеринбург : Профобразование, Уральский федеральный университет, 2019. — 92 с. — ISBN 978-5-4488-04014, 978-5-7996-2834-5. — Текст : электронный // Электронно-библиотечная

система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/87852.html. — Режим доступа: для авторизир.
пользователей

Дополнительная литература:

1. Федорова Г.И. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности. Учебное пособие. Изд.: КУРС, Инфра-М. Среднее профессиональное образование. 2016 г. 336 стр.