**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,
СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)**

Санкт-Петербургский колледж телекоммуникаций им. Э.Т. Кренкеля

УТВЕРЖДАЮ
Заместитель директора
по учебной работе
_____ Н.В. Калинина
17 марта 2022 г

СПбГУТ))) **Документ подписан простой электронной подписью**

Сертификат: 008a56eb36a1808f06
Владелец: Машков Георгий Михайлович
Действителен с 07.05.2022 по 06.05.2027

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ПРАКТИЧЕСКИХ РАБОТ

по учебной дисциплине
ОП.05. ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

по специальности
10.02.04 Обеспечение информационной безопасности телекоммуникационных систем

среднего профессионального образования

Санкт-Петербург
2022

ОП.05. Основы алгоритмизации и программирования. Методические указания по выполнению практических работ.

Составил: Обудовская А.А. – Санкт-Петербург, 2022.

Методические указания содержат описания практических занятий, предусмотренных рабочей программой Основы алгоритмизации и программирования. Каждая работа рассчитана на 2 академических часа, общий объём составляет 56 часов. Методические указания предназначены для обучающихся очной формы обучения по специальности 10.02.04 Обеспечение информационной безопасности телекоммуникационных систем.

Рассмотрено и одобрено предметной (цикловой) комиссией информационной безопасности телекоммуникационных систем Санкт-Петербургского колледжа телекоммуникаций им. Э.Т. Кренкеля.

ОГЛАВЛЕНИЕ

ЛАБОРАТОРНАЯ РАБОТА №1. Построение блок-схем с помощью программы Visio

Краткие сведения из теории

Алгоритм — это последовательность команд управления каким-либо исполнителем.
Совокупность величин, с которыми работает компьютер, принято называть *данными.* По отношению к программе различают *исходные, окончательные (результаты)* и *промежуточные* данные, которые получают в процессе вычислений. *Всякая величина занимает свое определенное место в памяти ЭВМ,* иногда говорят — ячейку памяти.

Любая величина имеет три основных свойства: *имя*, *значение* и *тип.* На уровне команд процессора величина идентифицируется адресом ячейки памяти, в которой она хранится. В алгоритмах и языках программирования величины подразделяются на константы и переменные.

Константа — неизменная величина, и в алгоритме она представляется собственным значением, например: 15, 34.7, *k,* True и др.

Переменные величины могут изменять свои значения в ходе выполнения программы и представляются в алгоритме символическими именами — идентификаторами, например: X, S2, cod 15 и др. Любые константы и переменные занимают ячейку памяти, а значения этих величин определяются двоичным кодом в этой ячейке.

Типы величин характеризуются множеством допустимых значений, множеством допустимых операций, формой внутреннего представления. Типы констант определяются по контексту (т.е. по форме записи в тексте), а типы переменных устанавливаются в описаниях переменных.

По структуре данные подразделяются на *простые* и *структурированные.* Для простых величин, называемых также скалярными, справедливо утверждение *одна величина — одно значение,* а для структурированных — *одна величина — множество значений.*

Основным элементарным действием в вычислительных алгоритмах является *присваивание значения переменной величине.*
Если значение константы определено видом ее записи, то переменная величина получает конкретное значение только в результате присваивания, которое может осуществляться двумя способами: с помощью команды присваивания и с помощью команды ввода.
Формат команды присваивания следующий:

переменная := выражение

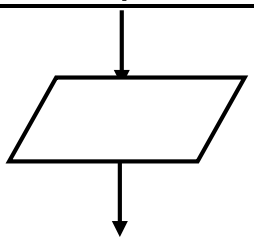
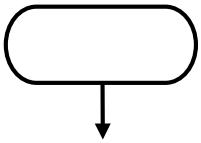
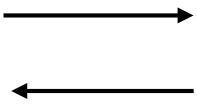
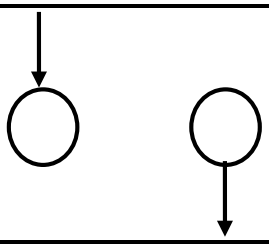
Знак «:=» следует читать как «присвоить».
Команда присваивания означает следующие действия, выполняемые компьютером:
1) вычисляется *выражение;*
2) полученное значение присваивается *переменной.*
При описании алгоритмов не обязательно соблюдать строгие правила записи выражений, это можно делать в обычной математической форме
Линейный алгоритм образуется из последовательности действий, следующих одно за другим.

При описании алгоритмов с помощью блок-схем типы данных, как правило, не указываются (но подразумеваются). В алгоритмическом языке для всех переменных типы данных указываются явно и их описание производится сразу после заголовка алгоритма. При этом используются следующие обозначения: **цел** — целые, **вещ** — вещественные, **лит** — символьные (литерные), **лог** — логические.

Общий вид алгоритма на алгоритмическом языке

алг название алгоритма (аргументы и результаты)
нач описание промежуточных величин
последовательность команд (тело алгоритма)
кон

Блочные символы линейной структуры

Название символа	**Обозначение**
Процесс	
Ввод – вывод	
Пуск-останов	
Направление потока	
Точки соединения	

При решении различных задач с помощью компьютера бывает необходимо вычислить логарифм или модуль числа, синус угла и т. д. Вычисления часто употребляемых функций осуществляются посредством подпрограмм, называемых **стандартными функциями,** которые заранее запрограммированы и встроены в

транслятор языка. Стандартные функции алгоритмического языка приведены в таблице 1.

В качестве аргументов функций можно использовать константы, переменные и выражения.

Примеры:

```
sin(3.05),  sin(x), sin(2*y+t/2), min(a,5), min(a+b, a*b)
```

Таблица 1

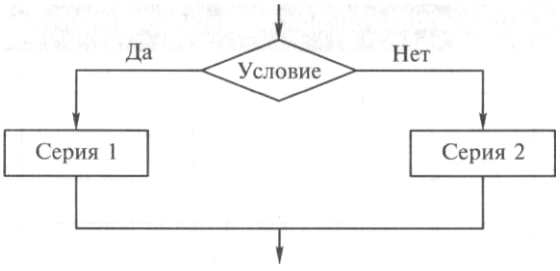
Название функции	**Математическое обозначение функции**	**Указатель функции**
абсолютная величина (модуль)	$\|x\|$	**abs(x)**
корень квадратный	$\sqrt{x}$	**sqrt(x)**
натуральный логарифм	ln x	**ln(x)**
десятичный логарифм	lg x	**lg(x)**
экспонента (степень числа е)	e^x	**exp(x)**
минимум из чисел x и y		**min(x,y)**
максимум из чиселх и y		**max(x,y)**
частное от деления из целых чисел x и y		**div(x,y)**
остаток от деления из целых чисел x и y		**mod(x,y)**
случайное число в диапазоне от 0 до		**rnd(x)**
синус	sin x	**sin(x)**
косинус	$\cos x$	**cos(x)**
тангенс	$tg\ x$	**tan(x)**
арктангенс	$arctg\ x$	**arctan(x)**

Арифметические выражения записываются по следующим правилам:

1. Нельзя опускать знак умножения между множителями и ставить рядом два знака **операций.**
2. Индексы элементов массивов записываются в квадратных скобках.
3. Для обозначения переменных используются буквы латинского алфавита.
4. **Операции выполняются в порядке старшинства:** сначала вычисление функций, затем возведение в степень, потом умножение и деление и в последнюю очередь сложение и вычитание.
5. **Операции одного старшинства выполняются слева направо.**

Базовая структура ВЕТВЛЕНИЕ. Обеспечивает в зависимости от результата проверки условия **(да** или **нет)** выбор одного из альтернативных путей работы алгоритма. Каждый из путей ведет к **общему выходу,** так что работа алгоритма будет продолжаться независимо от того, какой путь будет выбран.

Общий вид команды ветвления в блок-схемах и на алгоритмическом языке следующий:



если условие

то серия 1

иначе серия 2

всё

Сначала проверяется условие (вычисляется логическое выражение). Если условие истинно, то выполняется последовательность команд, на которую указывает стрелка с надписью «Да» (положительная ветвь) — «Серия 1». В противном случае выполняется отрицательная ветвь команд — «Серия 2».

На АЯ условие записывается после служебного слова «если», положительная ветвь — после слова «то», а отрицательная — после слова «иначе». Буквами «всё» в такой записи обозначают конец ветвления.

В записи логических выражений, помимо арифметических операций, используются операции отношения: < (меньше), >(больше), <= (меньше или равно), >= (больше или равно), = (равно), <> (не равно), а также логические операции – **И, ИЛИ, НЕ.**

Теоретические сведения о MS VISIO

Любой технологический процесс можно описать различными способами: текст, графическое отображение и др. Часто эти способы комбинируют и используют совместно в одном проекте. Наиболее наглядно и понятно представление в виде условной графики или рисунка. Однако для создания ряда схем, диаграмм существуют нормативные правила в виде государственных или международных стандартов и рекомендаций, например: для изображения алгоритмов, моделей процессов IDEF0 и IDEF3.

Для правильного построения подобных схем необходимо иметь в наличии соответствующую регламентирующую документацию, что не всегда выполнимо. В целях обеспечения унификации, быстроты и удобства исполнения, повышения качества представляемых графических изображений различных бизнес-схем и диаграмм, компанией Microsoft создана программа Microsoft Office Visio. В ряде случаев простая программа MS Visio может заменить дорогостоящие графические процессоры и системы визуального моделирования деловых процессов. Графические изображения, созданные в MS Visio, можно вставлять в виде объектов в файлы других программных продуктов Microsoft. В центральной части расположена область вставки, которая может содержать несколько страниц. Слева – окно «Фигуры» (Shapes), в котором отображаются выбранные пользователем трафареты с наборами фигур. Формат файлов MS Visio – *.vsd.

MS Visio позволяет преобразовать сложный текст и таблицы, которые трудно понять, в наглядные схемы, которые позволяют быстро донести информацию. Существует множество типов схем MS Visio, в том числе организаций, схемы сети, рабочий процесс, планы для дома и офиса. Начало работы с MS Visio можно обобщить в три основных этапа: *использование шаблона, организация и соединение фигур, а также изменение фигур с текстом.*

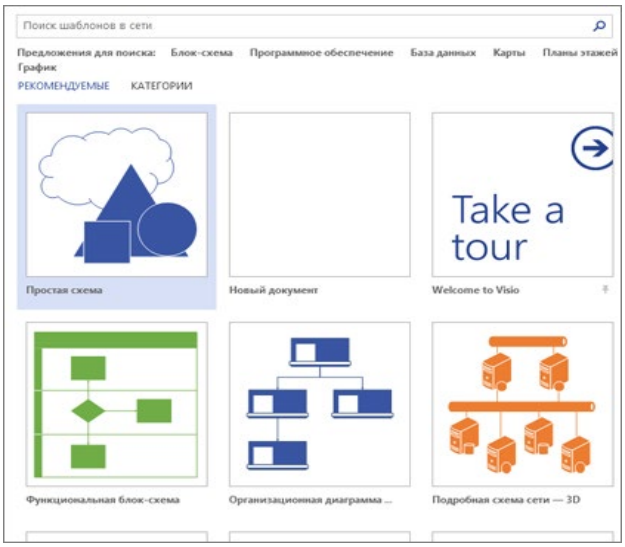
Выбор и открытие шаблона

Шаблоны включают в себя шаблоны, фигуры и размеры сетки, которые помогут вам быстро и легко при создании схемы начать работу.

Запустите MS Visio.

1. Выберите нужный шаблон или выберите "Базовая **схема",** чтобы начать с нуля.

На вкладке **Категории** доступны и другие шаблоны. Кроме того, вы можете воспользоваться функцией поиска шаблонов в Интернете.



Если вы используете ссылку на рабочий стол, может потребоваться указать определенный тип шаблона и выбрать **"Создать".**

Чтобы создать схему, перетащите фигуры из этого окна на полотно и соедините их. Существует несколько способов соединения фигур, но самый простой — с помощью стрелок автосоединения.

Фигуры Visio — это готовые объекты, которые можно перетаскивать на страницу схемы.

При перетаскивании фигуры из окна "Фигуры" на страницу документа исходная фигура остается в наборе. Она называется *образцом фигуры*. Фигура, помещенная в документ, является копией — так называемым *экземпляром* этого образца. В документ можно поместить сколько угодно экземпляров одной и той же фигуры.

Вращение и изменение размера фигур

Маркеры поворота

Круглый маркер  над выбранной фигурой называется маркером поворота. Чтобы повернуть фигуру влево или вправо, перетащите его в соответствующую сторону.

Стрелки для автосоединения

Соединительные стрелки помогают соединять фигуры друг с другом, как было описано в предыдущем разделе.

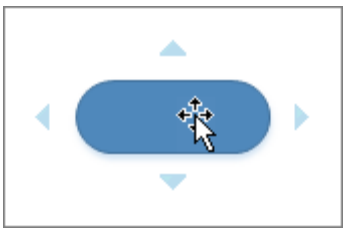
Маркеры выбора для изменения размера фигуры

Изменить высоту и ширину фигуры можно с помощью квадратных полосы выделения. Щелкните и перетащите угловой угол фигуры, чтобы увеличить фигуру, не изменяя ее пропорции, или щелкните и перетащите сбоку, чтобы сделать фигуру выше или шире.

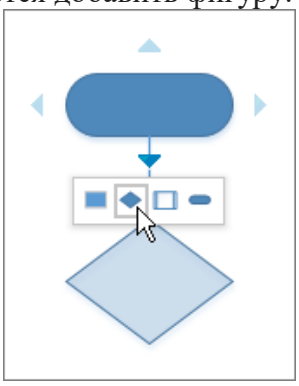
Автоматическое соединение фигуры

Один из способов соединения фигур состоит в том, чтобы позволить Visio автоматически создавать соединительные линии при добавлении фигур на страницу. Это особенно удобно при создании блок-схем.

1. Включите функцию автосоединения. Для этого на вкладке **Вид → Визуальные подсказки → Автосоединение**.
2. Перетащите фигуру из области **Фигуры** на страницу.
3. Наведите указатель мыши на исходную фигуру и удерживайте его, пока вокруг фигуры не появятся стрелки автосоединения.



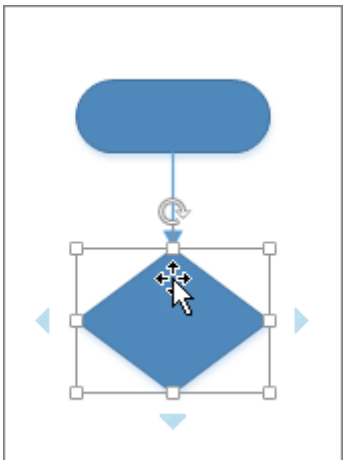
4. Наведите указатель мыши на стрелку, указывающую в том направлении, в котором требуется добавить фигуру.



Появится мини-панель инструментов с первыми четырьмя экспресс-фигурами, доступными в наборе элементов **Экспресс-фигуры**. Если навести указатель мыши на фигуру на этой панели инструментов, будет показано предварительное расположение новой фигуры на странице.

5. Щелкните фигуру, которую хотите добавить.

6. Если вы хотите продолжить добавление фигур, наведите указатель мыши на стрелку автосоединения рядом с новой фигурой и повторите действия.



Соединение фигур на странице

1. На вкладке **Главная** в группе **Инструменты** щелкните **Соединительная линия** или нажмите клавиши CTRL+3.

2. Щелкните фигуру и перетащите соединительную линию на другую фигуру.

3. Завершив создание соединения, на вкладке **Главная** в группе **Инструменты** щелкните **Указатель** или нажмите клавиши CTRL+1.

Добавление текста к фигурам и соединителю

1. На вкладке **Главная** в группе **Сервис** нажмите кнопку **Текст** .

2. Щелкните в любом месте страницы или перетащите его, чтобы создать текстовое поле нужного размера.

3. Введите текст.

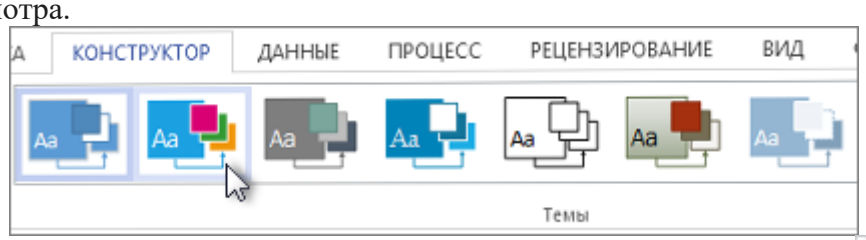
4. Повторите эти действия для всего текста, который вы хотите добавить.

5. Чтобы вернуться к обычному редактированию, на вкладке Главная в группе Инструменты нажмите кнопку Указатель .

Применение темы к схеме

1. На вкладке **Конструктор** попробуйте навести указатель мыши на разные темы.

При этом каждая из них будет временно применяться для предварительного просмотра.



2. Чтобы увидеть другие доступные темы, **нажмите** кнопку " ".

3. Щелкните нужную тему, чтобы применить ее к схеме.

Для выполнения отчетов лабораторных работ выбирайте «белую тему».

Задания: Составить алгоритмы в виде блок-схемы для заданий лабораторной работы при помощи MS VISIO, приведенных в вариантах. Написать псевдокод задания. Номер варианта указывает преподаватель.

Номер варианта	**Задание**
1	1. Даны стороны прямоугольника. Найти его периметр и длину диагонали. 2. $y=\begin{cases} atgx+\sqrt{\lvert x-2\rvert}, & x<2 \\ \left(a^2-b^2\right)\cdot\cos\pi x, & x=2 \\ (x-2)^3\cdot\sin\dfrac{\pi x}{2}, & x>2 \end{cases}$
2	1. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью. 2. $y=\begin{cases} \sin x+\sqrt{\lvert x-5\rvert}, & x<5 \\ a^2\cos\pi x+\ln(x+a), & x=5 \\ (x-5)^3\cdot tg\dfrac{x}{2}, & x>5 \end{cases}$
3	1. Дано ребро куба. Найти площадь грани, площадь боковой поверхности и объём куба. 2. $y=\begin{cases} ctgx+a\sqrt{\lvert x+2\rvert}, & x<-2 \\ \left(a^2-b^2\right)\cdot\cos\pi x, & x=-2 \\ (x-2)^3\cdot\sin\dfrac{\pi x}{3}, & x>-2 \end{cases}$
4	1. Найти площадь равнобедренной трапеции с основаниями a и b и высотой h.

Номер варианта	Задание
	2. $y=\begin{cases} ax+tgx+\sqrt{\ln\lvert x-4\rvert}, & x<-4 \\ \left(x^2-b^2\right)+\cos\pi x, & x=4 \\ (x-2)\cdot\sin 2\pi x, & x>4 \end{cases}$
5	1. Найти площадь кольца, внутренний радиус которого равен r, а внешний – заданному числу R (R > r). 2. $y=\begin{cases} \dfrac{1}{\sin x+2}, & если \quad x\le 0 \\ \lg x+e^x, & если \quad 0<x\le 2 \\ 2x^2, & если \quad x>2 \end{cases}$
6	1. Даны радиус основания конуса и его высота. Найти объем конуса. 2. $y=\begin{cases} e^x+\dfrac{1}{x+1}, & если \quad x<3 \\ \sin x+\sqrt{x}, & если \quad x=3 \\ \cos x+\lvert b\rvert, & если \quad x>3 \end{cases}$
7	1. Дана длина стороны равностороннего треугольника. Найти площадь этого треугольника и его высоту. 2. $y=\begin{cases} e^x+\dfrac{1}{x+1}, & если \quad x<4 \\ \sin x+\sqrt{x}, & если \quad x=4 \\ \cos x+\lvert b\rvert, & если \quad x>4 \end{cases}$
8	1. Составить алгоритм решения линейного уравнения $ax+b=0 \quad (a\ne 0)$ 2. $y=\begin{cases} \pi x^2+\lg x^2, & при \ x<1{,}5 \\ a+x, & при \quad x=1{,}5 \\ e^x+tgx, & при \quad x>1{,}5, \end{cases}$
9	1. Даны два действительных положительных числа *a* и *b.* Найти среднее арифметическое и среднее геометрическое этих чисел. 2. $y=\begin{cases} \pi x^2-9x^2, & при \quad x<1{,}4 \\ ax^3+17\sqrt{x}, & при \quad x=1{,}4 \\ \ln\left(x+11\sqrt{\lvert x+a\rvert}\right), & при \ x>1{,}4, \end{cases}$
10	1. Известны координаты на плоскости двух точек. Составить программу вычисления расстояния между ними.

Номер варианта	Задание
	2. $y=\begin{cases}\sqrt{te^{x/2}}, если & x\le 2\\ \lvert u\rvert e^{2x}, если & 2<x\le 6,\\ \sin^2(2t^3+3x), если & x>6\end{cases}$
11	1. Даны длины сторон основания и высота правильной четырёхугольной пирамиды. Найти её объём. 2. $y=\begin{cases}a\lg x+\sqrt[3]{\sin(x)}, при & x<1\\ 2a\cos x+e^x, при & 1\le x\le 2,\\ x^4+5a, при & x>2\end{cases}$
12	1. Даны длины сторон прямоугольного параллелепипеда. Найти полную площадь его поверхности. 2. $y=\begin{cases}\sin x\cdot\lvert\lg x\rvert, при & x<3{,}5\\ \cos^2 x+e^x, при & x=3{,}5,\\ 2x+5x^3, при & x>3{,}5\end{cases}$
13	1. Даны два числа $(a\ne b, b\ne 0)$. Найти их сумму, разность, произведение и частное. 2. $y=\begin{cases}\dfrac{\left(\ln^3 x+x\right)}{\sqrt{x+1}}, при & x<0{,}5\\ \sqrt{x+t}+e^x, при & x=0{,}5\\ \cos x+t\sin^2 x, при & x>0{,}5,\end{cases}$
14	1. Даны две стороны треугольника и угол между ними. Вычислить площадь треугольника. 2. $y=\begin{cases}bx-\lg bx, при & x>1\\ 1, при & x=1\\ \sin bx+\lvert bx\rvert, при & x<1,\end{cases}$
15	1. Даны диагонали ромба. Найти его периметр. 2. $y=\begin{cases}\lg(x+1), при & x<1{,}3\\ \sin^2\sqrt{ax}, при & x=1{,}3\\ 2x+5a, при & x>1{,}3\end{cases}$

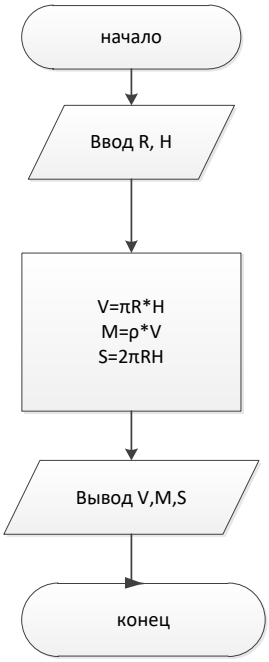
Методические указания

ПРИМЕР 1: Составить блок-схему и написать псевдокод для вычисления массы, объема и боковой поверхности медного цилиндра радиуса R см и высоты h см

Расчетные формулы для вычислений

$$V = \pi R^2 H$$
$$m = \rho \cdot V$$
$$S = 2\pi RH$$
$$\rho_{меди} = 8{,}9\,г/см^3$$

Алгоритм в виде блок-схемы выглядит следующим образом:



Запись алгоритма на алгоритмическом языке:

алг вычисления (**арг** **вещ** r, h; **рез** **вещ** v, m, s)

нач

ввод r, h

$v := \pi * r * r * h$

$m := \rho * v$

$s := 2 * \pi * r * h$

вывод v, m, s

кон

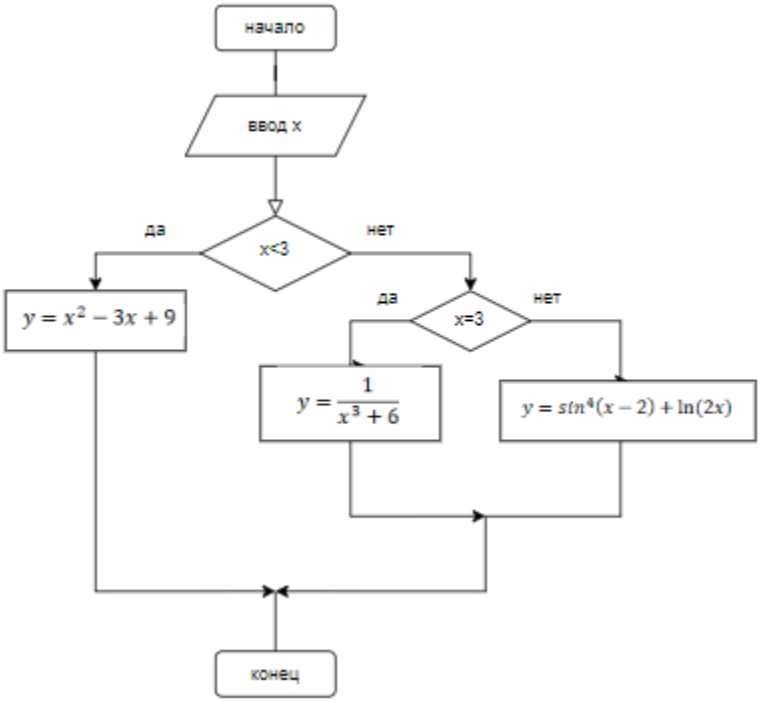
Пример 2: Вычислить значение функции. Зарисовать блок-схему, написать псевдокод.

$$y=\begin{cases} x^2-3x+9, & x<3 \\ \dfrac{1}{x^3+6}, & x=3 \\ \sin^4(x-2)+\ln(2x), & x>3 \end{cases}$$

Псевдокод и блок-схема задания выглядит следующим образом:

Псевдокод:

```
алг функция (арг вещ  x; рез вещ y)
нач
    ввод  x
    если x < 3
        то y := x * x − 3 * x + 9
    иначе
        если x=3
            то y := 1/(x * x * x + 6)
        иначе y:=pow(sin(x-2),4)+ln(2*x)
    всё
    вывод x, y
кон
```



Содержание отчета

1. Название и цель лабораторной работы
2. Условие задания
3. Анализ задания и расчетные формулы и чертеж, если это необходимо
4. Блок-схема при помощи MS VISIO и запись на псевдокоде
5. Контрольный тест для каждой ветви

Контрольные вопросы

1. Что называют алгоритмом?
2. Какие способы записи алгоритмов?
3. Как записывается команда присваивания?
4. Чем отличаются команды присваивания A:=B и B:=A?
5. Чем различаются простые и составные команды?
6. Приведите примеры простых и составных команд
7. Перечислите все служебные слова, используемые в командах ветвления

ЛАБОРАТОРНАЯ РАБОТА №2. ЛИНЕЙНЫЕ ВЫЧИСЛИТЕЛЬНЫЕ ПРОГРАММЫ

1. Краткие сведения из теории

1.1. 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. Вот несколько примеров арифметических выражений, записанных по правилам языка C++:

```
a+b          12.5-z          2*(X+Y)
x++          x+++b           --n*2          n*=1
```

Три первых выражения имеют традиционную форму для языков программирования высокого уровня, поэтому их смысл очевиден. Следующие четыре выражения специфичны для языка C++.

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.*

Арифметические операции. К арифметическим операциям относятся:

— вычитание или унарный минус;
+ сложение или унарный плюс;
* умножение;
/ деление;
% деление по модулю (остаток от деления);
++ 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.

Рассмотрим особенности выполнения операции деления. *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++`, `--a` или `a--`. Три следующих оператора дают один и тот же результат:

```
x=x+1;    ++x;    x++.
```

В языке Си имеются дополнительные операции присваивания, совмещающие присваивание с выполнением других операций. Среди них: *+=, -=, /=, *=, %=*. Приоритет у них такой же, как и у простого присваивания. Примеры использования этих операций:

```
a+=2    эквивалентно a=a+2,
x-=a+b  эквивалентно x=x-(a+b),
p/=10   эквивалентно p=p/10,
m*=n    эквивалентно m=m*n,
r%=5    эквивалентно r=r%5.
```

Заметим, что вместо выражения a=a+2 предпочтительнее писать в программе a+=2, поскольку второе выражение будет вычисляться быстрее.

Приведение типов при вычислении выражений. Практически во всех языках программирования высокого уровня работает ряд общих правил записи выражений:

• все символы, составляющие выражение, записываются в строку (нет надстрочных и подстрочных символов);

• в выражении проставляются все знаки операций;

• при записи выражения учитываются приоритеты операций;

• для влияния на последовательность операций используются круглые скобки.

1.2. Линейные программы на C/C++

Структура программы. Общая структура программы на C/C++ следующая:

```
директивы_препроцессора
определение_функции_1
определение_функции_2
определение_функции_N
```

Среди функций обязательно присутствует главная функция с именемmain. Простейшая программа содержит только главную функцию и имеет следующую структуру:

```
директивы препроцессора
void main()
{ определения_объектов;
исполняемые_операторы;
}
```

Математические функции (заголовочный файл math.h)

Обращение	Тип аргумента	Тип результата	Функция
abs(x)	int	int	абсолютное значение целого числа
acos(x)	double	double	арккосинус (радианы)
asin(x)	double	double	арксинус (радианы)
atan (x)	double	double	арктангенс (радианы)
ceil (x)	double	double	ближайшее целое, не меньшее *x*
cos(x)	double	double	косинус (*x* в радианах)
exp(x)	double	double	e^x - экспонента от *x*
fabs(x)	double	double	абсолютное значение вещественного *x*

floor(x)	double	double	наибольшее целое, не превышающее *x*
fmod(x,y)	double /double	double	остаток от деления нацело *x* /y
log(x)	double	double	логарифм натуральный — ln x
log10(x)	double	double	логарифм десятичный — lg x
pow(x,y)	double double	double	*x* в степени *y* — x^y
sin(x)	double	double	синус (*x* в радианах)
sqrt(x)	double	double	корень квадратный (положительное значение)
tan(x)	double	double	тангенс (*x* в радианах)

Потоковый ввод-вывод в C++. Программируя на языке C++, можно пользоваться средствами ввода-вывода стандартной библиотеки C, подключаемой с помощью заголовочного файла `stdio.h`. Однако в C++ имеются свои 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;
```

Элемент вывода `endl` является так называемым 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. Тип данных и необходимый формат определяются автоматически.

2. Порядок выполнения работы

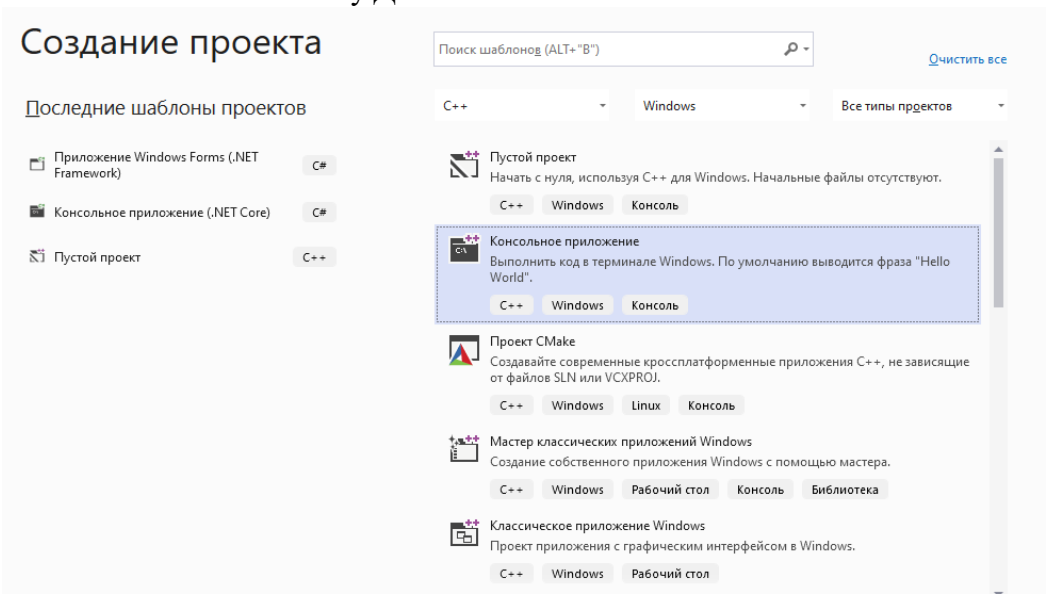
2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.

5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 **Указания для создания проекта в среде программирования Visual C++**

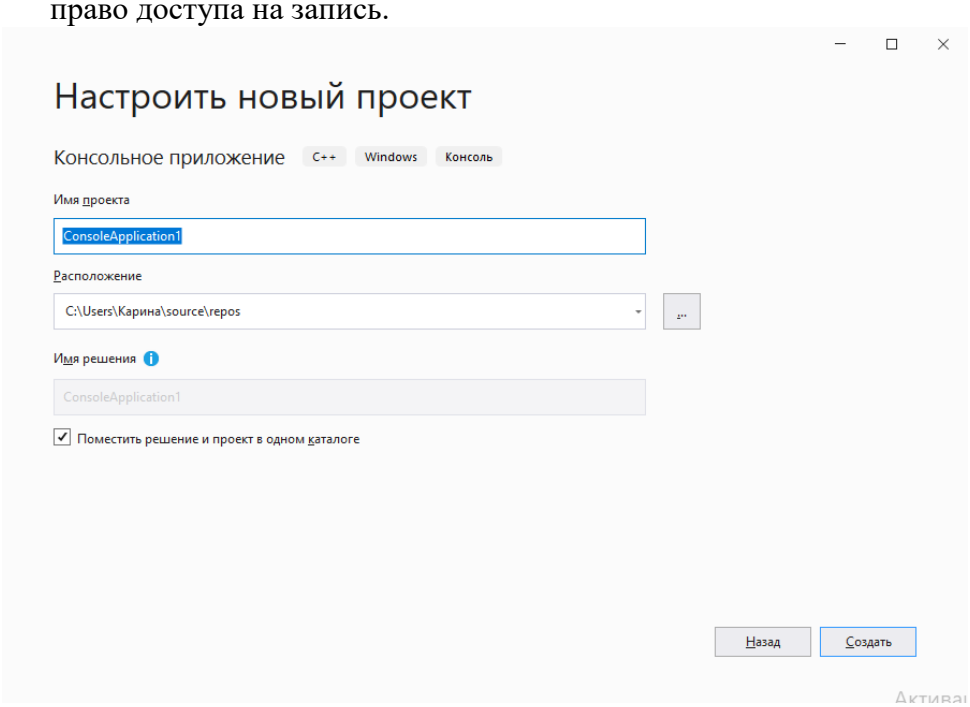
Для написания простейшей программы в среде разработки приложений Visual C++ необходимо в открывшемся окне «**Создание проекта**», выбрать пункт **Консольное приложение** и нажать кнопку **Далее**



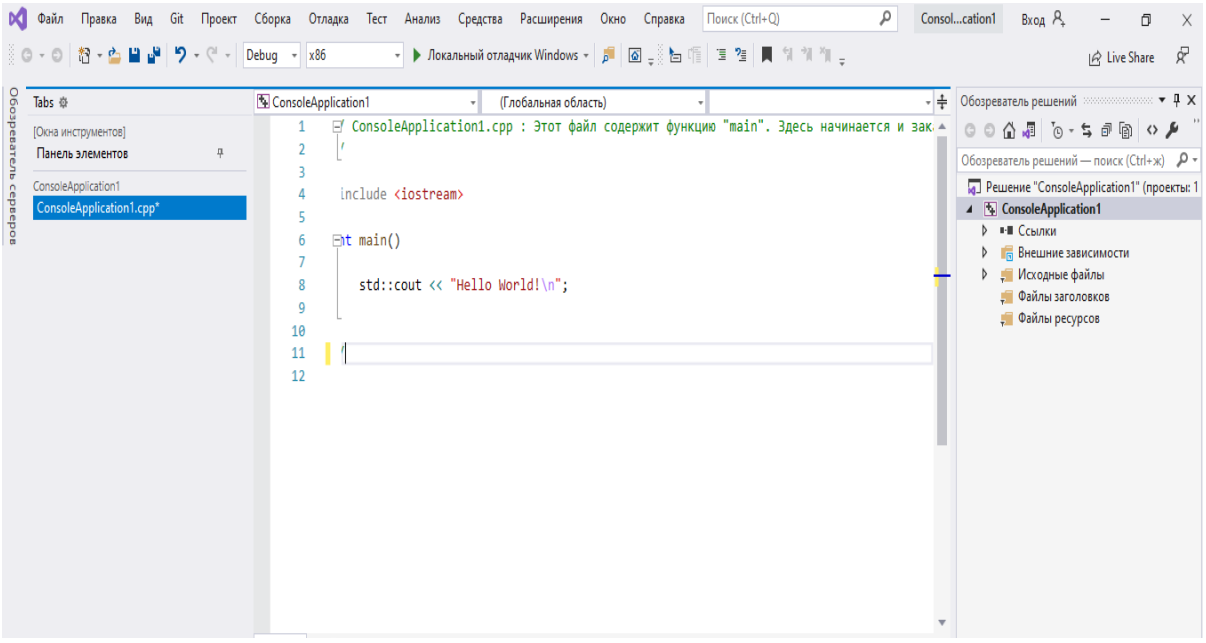
Откроется окно **Настроить новый проект**, в котором нужно указать Имя проекта и место его сохранения.

Здесь следует руководствоваться правилами:

1) Имя проекта указывается латинскими буквами и не может начинаться с цифры;
2) Расположение проекта – любая директива на диске, к которой пользователь имеет право доступа на запись.



После указания данных о проекте и нажатия кнопки **Создать**, будет открыто окно.



После этого будет создан и открыт пустой файл с расширением *.cln. В этом окне следует писать код программы.

Компиляция и запуск приложения осуществляются по нажатию на кнопку **F5**.

2.3 Рекомендации для составления кода программы на языке C++

Код программы реализует алгоритм расчёта по двум формулам в соответствии с индивидуальным заданием (результаты вычислений по обеим формулам должны совпадать). При вводе данных нужно учесть, что в программной среде предполагается измерение углов в радианах, а пользователь вводит углы в градусах. В связи с этим все углы следует после ввода исходных данных переводить из градусов в радианы по формуле: $\alpha = \alpha * \pi / 180$.

2.4 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения (общая часть, если таковая имеется, и индивидуальное задание).
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Если использованы визуальные компоненты, то рисунок формы.
6. Блок-схема программы (или обработчиков событий при использовании визуальных компонентов)
7. Результат тестирования программы.

3. Задание лабораторной работы № 2

Напишите программу для расчета по двум формулам в соответствии с индивидуальным заданием (результаты вычислений по обеим формулам должны совпадать).

вариант	задание	вариант	задание
1	$z_1 = 2\sin^2(3\pi - 2\alpha)\cos^2(5\pi + 2\alpha)$ $z_2 = \frac{1}{4} - \frac{1}{4}\sin\left(\frac{5}{2}\pi - 8\alpha\right).$	**2**	$z_1 = \cos\alpha + \sin\alpha + \cos 3\alpha + \sin 3\alpha$ $z_2 = 2\sqrt{2}\cos\alpha \cdot \sin\left(\frac{\pi}{4} + 2\alpha\right)$

3	$z_1 = \dfrac{\sin 2\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha + 1 - 2\sin^2 2\alpha}$ $z_2 = 2\sin\alpha$.	**4**	$z_1 = \dfrac{\sin\alpha + \sin 5\alpha - \sin 3\alpha}{\cos\alpha - \cos 3\alpha + \cos 5\alpha}$ $z_2 = tg3\alpha$.
5	$z_1 = 1 - \dfrac{1}{4}\sin^2 2\alpha + \cos 2\alpha$ $z_2 = \cos^2\alpha + \cos^4\alpha$.	**6**	$z_1 = \cos\alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha$ $z_2 = 4\cos\dfrac{\alpha}{2}\cdot\cos\dfrac{5\alpha}{2}\cdot\cos 4\alpha$.
7	$z_1 = \cos^2\left(\dfrac{3}{8}\pi - \dfrac{\alpha}{4}\right) - \cos^2\left(\dfrac{11}{8}\pi + \dfrac{\alpha}{4}\right)$ $z_2 = \dfrac{\sqrt{2}}{2}\sin\dfrac{\alpha}{2}$.	**8**	$z_1 = \cos^4 x + \sin^2 y + \dfrac{1}{4}\sin^2 2x - 1$ $z_2 = \sin(x+y)\cdot\sin(y-x)$.
9	$z_1 = (\cos\alpha - \cos\beta)^2 - (\sin\alpha - \sin\beta)^2$ $z_2 = -4\sin^2\dfrac{\alpha-\beta}{2}\cdot\cos(\alpha+\beta)$	**10**	$z_1 = \dfrac{\sin\left(\dfrac{\pi}{2} + 3\alpha\right)}{1 - \sin(3\alpha - \pi)}$ $z_2 = ctg\left(\dfrac{5}{4}\pi + \dfrac{3}{2}\alpha\right)$.
11	$z_1 = \dfrac{1 - 2\sin^2\alpha}{1 + \sin 2\alpha}$ $z_2 = \dfrac{1 - tg\alpha}{1 + tg\alpha}$.	**12**	$z_1 = \dfrac{\sin 4\alpha}{1 + \cos 4\alpha}\cdot\dfrac{\cos 2\alpha}{1 + \cos 2\alpha}$ $z_2 = ctg\left(\dfrac{3}{2}\pi - \alpha\right)$.
13	$z_1 = \dfrac{\sin\alpha + \cos(2\beta - \alpha)}{\cos\alpha - \sin(2\beta - \alpha)}$ $z_2 = \dfrac{1 + \sin 2\beta}{\cos 2\beta}$.	**14**	$z_1 = \dfrac{\cos\alpha + \sin\alpha}{\cos\alpha - \sin\alpha}$ $z_2 = tg2\alpha + \sec 2\alpha$.
15	$z_1 = \left(1 + ctg^2\alpha + \dfrac{1}{ctg^2\alpha}\right)$ $z_2 = \dfrac{1}{\sin^2\alpha\cdot\cos^2\alpha}$		

4. Пример выполнения лабораторной работы

4.1Условие задачи.

Написать программу для расчета по двум формулам (результаты вычислений по обеим формулам должны совпадать).

$$z_1 = \frac{2\cos^2\alpha - 1}{1 - \sin 2\alpha} - \frac{\cos\alpha - \sin\alpha}{\cos\alpha + \sin\alpha}$$

$$z_2 = \frac{4tg\alpha}{1 - tg^2\alpha}$$

Краткий список математических функций приведен в описании данной лабораторной работы. Для их использования необходимо подключить к программе заголовочный файл <math.h>. При необходимости можно обратиться к компьютерной справке. Для этого нужно подвести курсор к слову <math.h> и нажать одновременно клавиши Ctrl + F1.

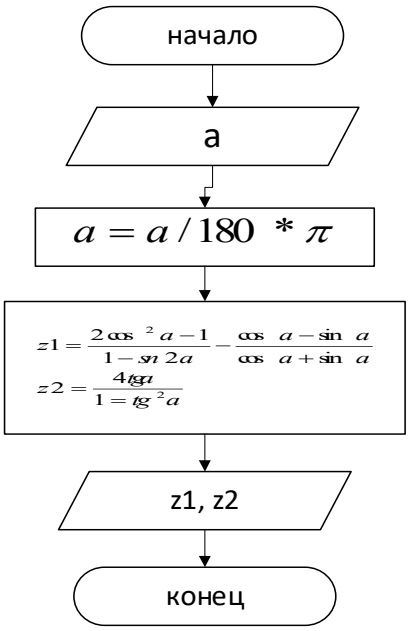
4.2. Ограничения на значения параметров

Знаменатель дроби не должен обращаться в ноль. Пока мы изучаем линейные программы и не можем предусмотреть обработку неверных исходных данных в коде программы. Поэтому на данном этапе работы нужно подбирать исходные данные (угол α) так, чтобы знаменатели дробей не были равны нулю.

4.3.Текст программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#define _USE_MATH_DEFINES // для работы с математическими
константами
#include<math.h>
using namespace std;  // использовать пространство имён std
int main(){           // заголовок главной функции
     setlocale(LC_ALL,".1251");    //  возможность  переключения на
русский язык
     float a,z1,z2;                //  описание типов переменных
     cout<<"Введите угол альфа a="; // вывод на экран сообщения
     cin>>a;                        // ввод величины угла
     a=a/180*M_PI;                  // перевод величины угла из
                                    // градусов в радианы
     z1=(2*cos(a)*cos(a)-1)/(1-sin(2*a))-
        (cos(a)-sin(a))/(cos(a)+sin(a)); // вычисление z1
     z2=4*tan(a)/(1-tan(a)*tan(a));         //  вычисление z2
     cout<<"z1="<<z1<<"\nz2="<<z2<<"\n";    //  вывод на экран
                                            //результатов
     system("PAUSE");                       //  задержкаэкрана
}
```

4.4. Блок-схема программы



4.5.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

```
Введите угол альфа a=0
```

```
z1=0
z2=0
Для продолжения нажмите любую клавишу…

Введите угол альфа a=30
z1=3.4641
z2=3.4641
Для продолжения нажмите любую клавишу…

Введите угол альфа a=22.5
z1=2
z2=2
Для продолжения нажмите любую клавишу…
```

Лабораторная работа № 3. ЛИНЕЙНЫЕ ПРОГРАММЫ ДЛЯ РЕШЕНИЯ ГЕОМЕТРИЧЕСКИХ И ФИЗИЧЕСКИХ ЗАДАЧ

1. Краткие сведения из теории

Пока мы будем составлять только простейшие программы. В качестве опорного примера рассмотрим программу для вычисления площади треугольника по формуле Герона.

Пример. Дано: *a, b, c* — стороны треугольника. Вычислить *S* — площадь треугольника. По формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)},$$

где *p* — полупериметр треугольника.

```
   #include <iostream.h>   //   библиотека ввода-вывода
   #include <math.h>
   #include <locale.h>     // для переключения на русский язык
using namespace std;  // использовать пространство имён std
void main ()  // заголовок главной функции
{setlocale(LC_ALL,".1251"); // возможность переключения на русский язык
   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)<<endl;
system("PAUSE");  //   задержка экрана

   }
```

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения (общая часть, если таковая имеется, и индивидуальное задание).
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Если использованы визуальные компоненты, то рисунок формы.
6. Блок-схема программы (или обработчиков событий при использовании визуальных компонентов)
7. Результат тестирования программы.

3. Вопросы к защите

1. Какие типы данных использованы в программе и почему?

2. Какие три простейшие алгоритмические структуры используются при написании программ?
3. Что такое "идентификатор"?
4. Что такое "директива препроцессора" и для чего она используется?
5. Какие операторы ввода-вывода использованы в программе и почему?
6. Зависит ли тип результата от типа исходных данных? И если зависит, то как именно?

4. **Задание лабораторной работы № 3**

Список индивидуальных заданий лабораторной работы

1. Вычислить периметр и площадь прямоугольного треугольника по длинам *a* и *b* двух катетов.
2. Заданы координаты трех вершин треугольника $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Найти его периметр и площадь.
3. Вычислить длину окружности и площадь круга одного и того же заданного радиуса *R*.
4. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
5. Вычислить расстояние между двумя точками с данными координатами (x_1, y_1) и (x_2, y_2).
6. Даны два действительных числа *x* и *y*. Вычислить их сумму, разность, произведение и частное.
7. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
8. Известна длина окружности. Найти площадь круга, ограниченного этой окружностью.
9. Найти площадь кольца, внутренний радиус которого равен *r*, а внешний – *R (R>r)*.
10. Найти площадь треугольника, две стороны которого равны *a* и *b*, а угол между этими сторонами γ.
11. Вычислить площадь трапеции по заданным основаниям и высоте.
12. Вычислить высоты треугольника со сторонами *a, b, c*.
13. Составить программу вычисления объема цилиндра и конуса, которые имеют одинаковую высоту *H* и одинаковый радиус основания *R*.
14. Вычислить объем и площадь поверхности цилиндра с заданным радиусом основания и высотой.
15. Даны длины сторон параллелограмма и угол между ними. Найти площадь параллелограмма.

5. Пример выполнения лабораторной работы

5.1.Условие задачи.

Дано действительное число *R* вида nnn.ddd (три цифровых разряда в дробной и целой частях). Поменять местами дробную и целую части числа и вывести полученное значение числа.

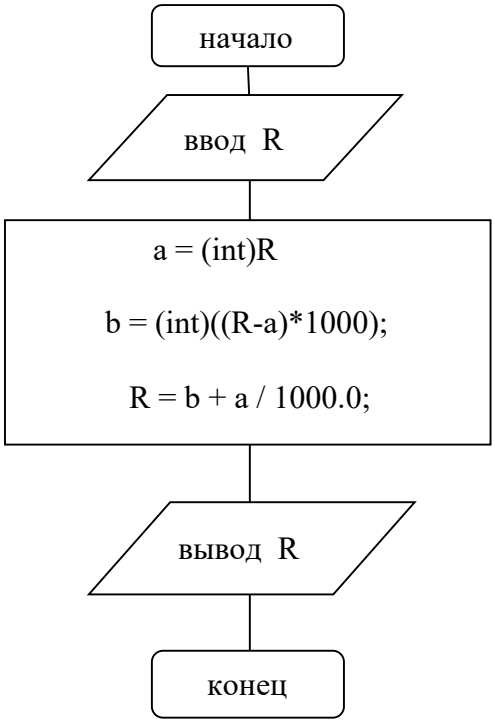
5.2.Ограничения на значения параметров

На данном этапе мы изучаем линейные программы и не можем предусмотреть обработку неверных исходных данных в коде программы. Поэтому нужно задавать число R строго в заданном виде (три цифровых разряда в дробной и целой частях).

5.3.Текст программы

```cpp
#include <iostream>
#include<locale.h>  // для переключения на русский язык
using namespace std;
int main()
{
    setlocale(LC_ALL, ".1251");  //  возможность  переключения на русский язык
    double R;
    int a, b;

    cout<<"Введите число R = ";
    cin >> R;

    a = (int)R;                  // находим целую часть числа
    b = (int)((R - a) * 1000);   // находим дробную часть числа
    R = b + a / 1000.0;   // целую и дробную части числа меняем местами
    cout << "\nR = " << R;
}
```

5.4. Блок-схема программы



5.5.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

```
Введите число R = 123.456

R = 456.123
```

Лабораторная работа № 4. ПРОГРАММЫ ДЛЯ РЕШЕНИЯ ЗАДАЧ С ИСПОЛЬЗОВАНИЕМ УСЛОВНОГО ОПЕРАТОРА

1. Краткие сведения из теории

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`. Блок-схема полной формы условного оператора выглядит следующим образом:

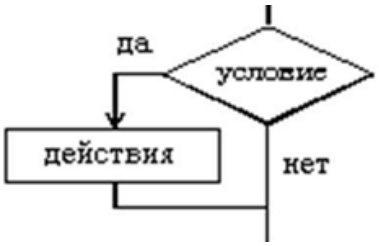
Необходимо обратить внимание на следующие особенности синтаксиса условного оператора:

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

Возможно использование неполной формы условного оператора

```
if (выражение) оператор;
```

Блок-схема в этом случае имеет вид:



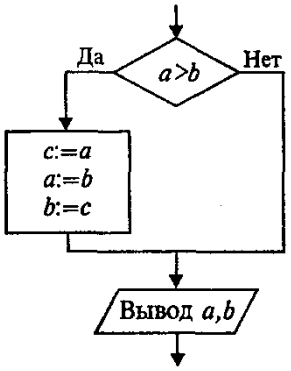
Вот пример использования полной формы условного оператора для нахождения большего значения из двух переменных *a* и *b:*

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

Та же самая задача может быть решена с использованием неполного ветвления следующим образом:

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

Пример 1. 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. В Си фигурные скобки выполняют роль операторных скобок по аналогии с **Begin, End** в Паскале.

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

В следующем примере вернемся к задаче вычисления площади треугольника по длинам трех сторон. Добавим в программу проверку условия правильности исходных данных: *a*, *b, c* должны быть положительными, а сумма длин каждой пары сторон треугольника должна быть больше длины третьей стороны.

Пример 2.

```
// Площадь треугольника
#include <iostream.h>
#include <math.h>
void main()
{float a,b,c,p,s;
cout<<"\na="; cin>>a;
cout<<"\nb="; cin>>b;
cout<<"\nc="; cin>>c;
if(a>0 && b>0 && c>0 && a+b>c && a+c>b && b+c>a)
{ p=(a+b+c)/2;
  s=sqrt(p*(p-a)*(p-b)*(p-c));
  cout<<"\nПлощадь треугольника="<<s;
}
    else cout("\n Неверные исходные данные.");
}
```

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы (или обработчиков событий при использовании визуальных компонентов)
6. Результат тестирования программы.

3. **Вопросы к защите**

1. Для чего используется условный оператор?
2. Что такое "неполная форма условного оператора"?
3. Как организовать выполнение нескольких операторов при заданном условии?
4. Что такое "вложенное условие"?
5. Каким образом отражается условный оператор в блок-схеме?

4. **Индивидуальные задания лабораторной работы**

1. Даны три действительных числа. Возвести в квадрат те из них, значения которых неотрицательны, и в четвертую степень — отрицательные.
2. Подсчитать количество отрицательных среди чисел *a, b, c.*
3. Составить программу, осуществляющую перевод величин из метров в километры и наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнять указанное действие.
4. Определить, делителем каких чисел *a, b, c* является число *k.*
5. Даны действительные числа *a, b, c*. Удвоить эти числа, если $a \geq b \geq c$, и заменить их абсолютными значениями, если это не так.
6. Составить программу, осуществляющую перевод величин из радианной меры в градусную и наоборот. Программа должна запрашивать, какой перевод нужно осуществить, и выполнять указанное действие.
7. Подсчитать количество положительных среди чисел a, *b, c.*
8. Написать программу — модель анализа пожарного датчика в помещении, которая выводит сообщение «Пожароопасная ситуация», если температура в комнате превысила 60° С.
9. Рис расфасован в два пакета. Масса первого — *m* кг, второго — *n* кг. Составить программу, определяющую, какой пакет тяжелее — первый или второй.
10. К финалу конкурса лучшего по профессии «Специалист электронного офиса» были допущены трое: Иванов, Петров, Сидоров. Соревнования проходили в два тура. Иванов в первом туре набрал m_1 баллов, во втором — n_1. Петров — m_2, n_2 соответственно; Сидоров — m_3, n_3. Составить программу, определяющую, сколько баллов набрал победитель.
11. Составить программу, которая проверяла бы, не приводит ли суммирование двух целых чисел *A* и *B* к переполнению (т.е. к результату большему чем 32 767). Если будет переполнение, то сообщить об этом, иначе вывести сумму этих чисел.
12. Написать программу, определяющую, будут ли прямые
    $A_1x + B_1y + C_1 = 0$ и $A_2x + B_2y + C_2 = 0$ перпендикулярны.
    Условие перпендикулярности прямых: $A_1A_2 + B_1B_2 = 0$.
13. Подсчитать количество целых среди чисел *a*, *b, c.*
14. В ПК поступают результаты соревнований по плаванию для двух спортсменов. Составить программу, которая выбирает лучший результат и выводит его на экран с

сообщением, что это результат победителя заплыва.

15. Написать программу, вычисляющую частное от деления двух чисел. Если в знаменателе ноль, программа должна выдавать сообщение об этом.

5. Пример выполнения лабораторной работы

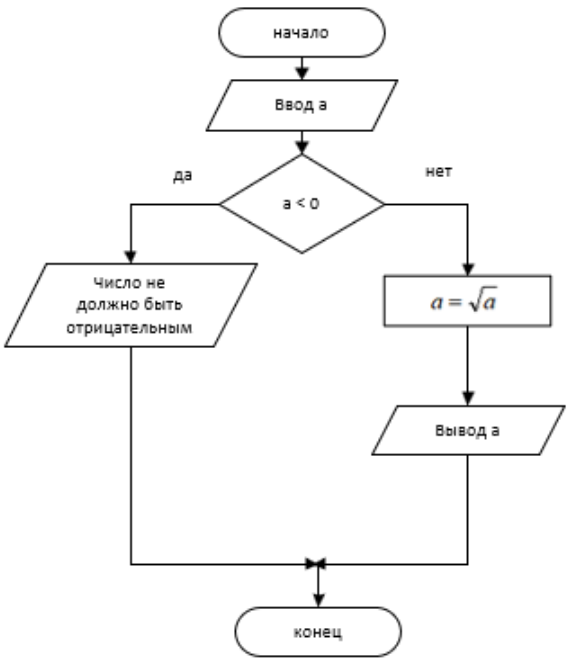
5.1.Условие задачи.

Написать программу вычисления квадратного корня из заданного числа. Если задано отрицательное число, программа должна выводить сообщение об этом.

5.2.Текст программы

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык
 using namespace std;  // использовать пространство имён std

int main() {           // заголовок главной функции
    setlocale(LC_ALL, ".1251");    //  возможность  переключения на русский язык
    float a;        //  описание типов переменных
    cout << "  Введите число  a = "; cin >> a;
    if (a < 0)cout << "\n  Число не должно быть отрицательным\n\n";
    else {
        a = sqrt(a);
        cout << "\n  Результат:  " << a << "\n\n"; //вывод результата
    }

}
```

5.3.Блок-схема программы



5.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

1)
```
Введите число  a = 22
Результат:  4.69042
```

2) Введите число a = -22

Число не должно быть отрицательным

ЛАБОРАТОРНАЯ РАБОТА №5. ПРОГРАММЫ ДЛЯ ИССЛЕДОВАНИЯ ОБЛАСТЕЙ, ОПИСЫВАЕМЫХ ЛОГИЧЕСКИМИ ВЫРАЖЕНИЯМИ

1. Краткие сведения из теории

1.1. Операции отношения.

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

Как уже говорилось раньше, в стандарте С++ нет логического типа данных. Поэтому 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`.

1.2.Логические операции. Три основные логические операции в языке С++ записываются следующим образом:

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

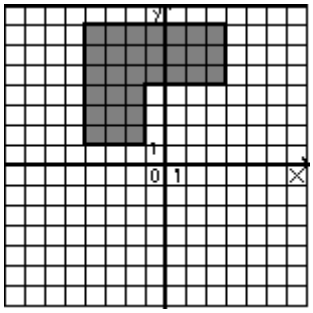
Правила их выполнения определяются таблицей истинности.
Например, логическое выражение, соответствующее системе неравенств $0 < x < 1$ в программе на С++ запишется в виде следующего логического выражения:

```
x>0&& x<1
```

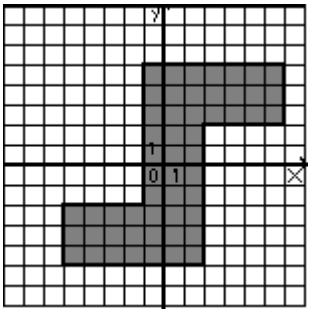
2. Задание лабораторной работы

Составить блок-схему алгоритма и написать программу, которая вводит координаты точки (***x***, ***y***) и определяет, попадает ли точка в заштрихованную область на рисунке, который соответствует Вашему варианту. Попадание на границу области считать попаданием в область.

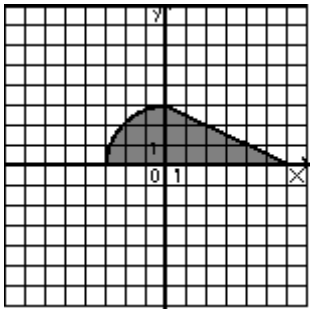
Варианты индивидуальных заданий



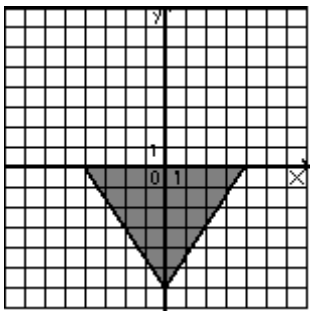
1.



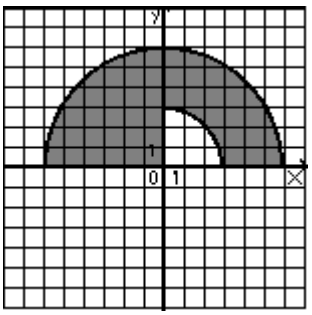
2.



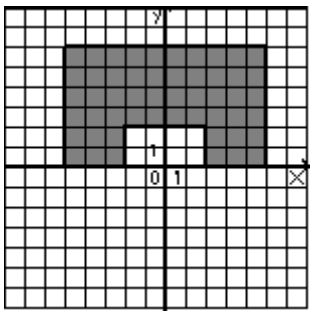
3.



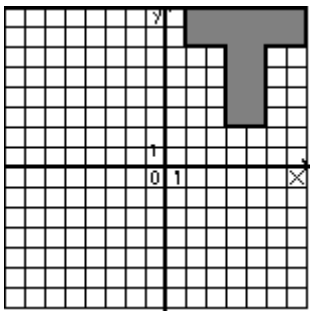
4.



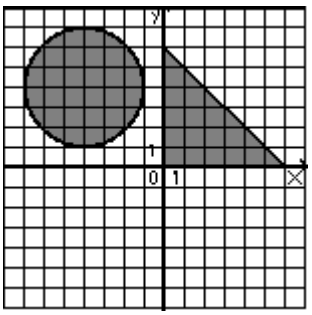
5.



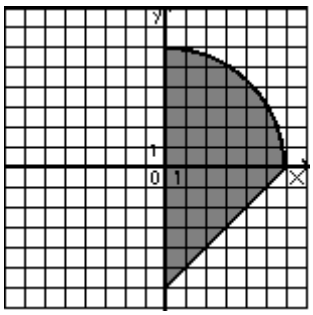
6.



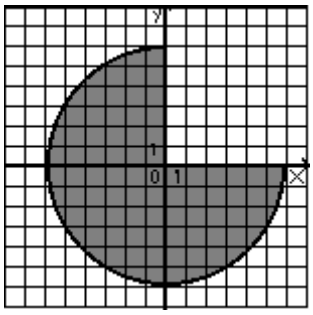
7.



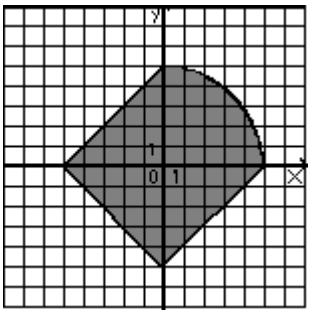
8.



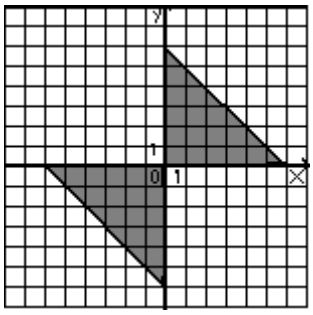
9.



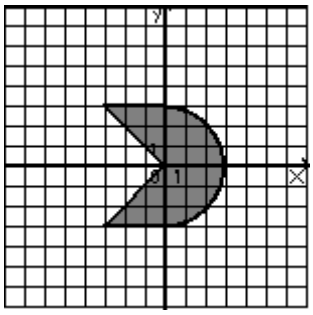
10.



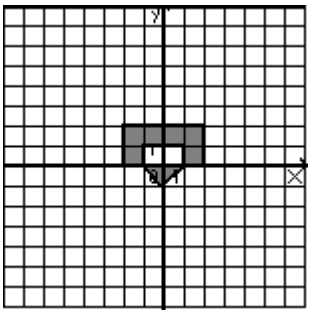
11.



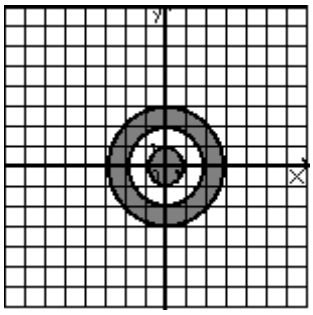
12.



13.



14.



15.

3. Порядок выполнения работы

3.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

3.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы (или обработчиков событий при использовании визуальных компонентов)
6. Результат тестирования программы.

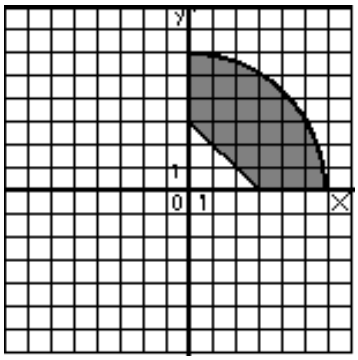
4. Вопросы к защите

1. Перечислите операции отношения и приведите их запись в C++.
2. Перечислите логические операции. Как они записываются в C++?
3. В каком порядке выполняются в C++ логические операции и операции отношения?
4. Как составить уравнение прямой, проходящей через две заданные точки?
5. Напишите уравнение окружность заданного радиуса с центром в начале координат.

5. Пример программы

5.1.Условие задачи.

Написать программу, которая вводит координаты точки (***x***, ***y***) и определяет, попадает ли точка в заштрихованную область на рисунке. Попадание на границу области считать попаданием в область.



5.2.Разработка кода программы.

Обозначим *x, y* – координаты точки. После ввода значений этих переменных нужно разработать условие, описывающее заданную область. В нашем случае область на плоскости ограничена четырьмя линиями. Тремя отрезками прямой и дугой окружности. Уравнение окружности в общем случае имеет вид $x^2 + y^2 = r^2$. У нас радиус равен 6, и рассматриваемая область лежит внутри окружности. Следовательно, одно из условий будет иметь вид:
$x^2 + y^2 <= 36$.

Область ограничена также наклонной прямой, проходящей через точки с координатами (0; 3) и (3; 0). Для составления уравнения можно воспользоваться уравнением прямой с угловым коэффициентом: $y = k \cdot x + c$.Подставив вместо *x* и *y* известные координаты, получим систему двух уравнений с двумя неизвестными *k* и *c*. Решив систему уравнений и подставив найденные коэффициенты в уравнение прямой, получим: $y = -x + 3$. Так как рассматриваемая область находится выше прямой, то нужно использовать неравенство

$y >= -x + 3$.

Оставшиеся две линии, ограничивающие область – это координатные оси. В первой четверти должно быть $x >= 0$ и $y >= 0$. Все четыре условия должны быть объединены логическим "И", так как выполняются одновременно. Окончательно получим следующее условие:

`(x² + y²<=36&&y>= –x + 3&&x>= 0&&y>= 0).`

После проверки условия, нужно вывести на экран соответствующее сообщение.

5.3.Текст программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#define _USE_MATH_DEFINES // для работы с математическими
константами
#include<math.h>
using namespace std;  // использовать пространство имён std
void main(){              // заголовок главной функции
     setlocale(LC_ALL,".1251"); // возможность переключения на
русский язык
     float x,y;          //  описание типов переменных
     cout<<"Введите координаты точки \nx=";
     cin>>x;
     cout<<"y=";
     cin>>y;
     if (x>=0 && y>=0 && y>=-x+3 && x*x+y*y<=36)
          cout<<"\nТочка попадает в заданную область\n";
     else cout<<"\nТочка не попадает в заданную область\n";
     system("PAUSE");                         //  задержка экрана
}
```

5.4.Тестирование программы

Самое важное в отладке этой программы - убедиться в том, что программа выдает правильные результаты при разных комбинациях входных данных. Следовательно, необходимо подобрать такие комбинации, которые были бы показательными для разных случаев размещения точки. Мы предлагаем такие комбинации:

1) точка лежит ниже области - (4, -0.5);
2) точка лежит на нижней границе области - (4, 0);
3) точка лежит внутри области - (3, 3);
4) точка лежит на наклонной прямой - (1, 2);
5) точка лежит выше области - (3,7);

При выбранных начальных данных получим:

```
Введите координаты точки
x=4
y=-0.5
Точка не попадает в заданную область
Для продолжения нажмите любую клавишу…

Введите координаты точки
x=4
y=0
```

```
Точка попадает в заданную область
Для продолжения нажмите любую клавишу…

Введите координаты точки
x=3
y=3
Точка попадает в заданную область
Для продолжения нажмите любую клавишу…

Введите координаты точки
x=1
y=2
Точка попадает в заданную область
Для продолжения нажмите любую клавишу…

Введите координаты точки
x=3
y=7
Точка не попадает в заданную область
Для продолжения нажмите любую клавишу…
```

ЛАБОРАТОРНАЯ РАБОТА № 6. ВЛОЖЕННЫЕ УСЛОВНЫЕ ОПЕРАТОРЫ

1. Краткие сведения из теории

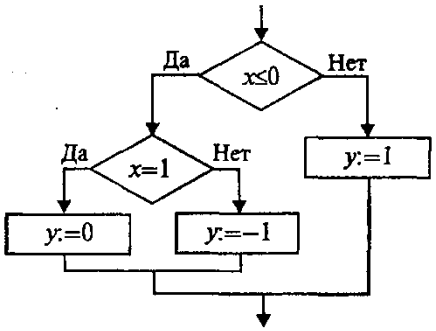
Условный оператор. Формат условного оператора следующий:

```
if (выражение) оператор1; else оператор2;
```

Теперь рассмотрим примеры программирования вложенных ветвящихся структур. Требуется вычислить функцию sign(x) — знак x, которая определена следующим образом:

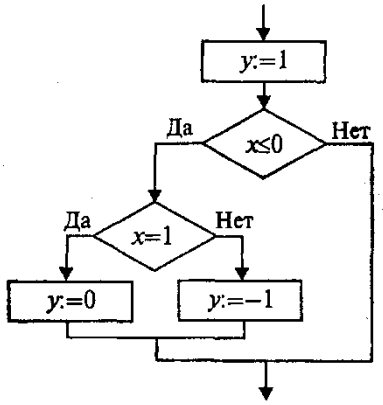
$$\text{sign}(x) = \begin{cases} -1, & \text{если } x < 0, \\ 0, & \text{если } x = 0, \\ 1, & \text{если } x > 0. \end{cases}$$

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 ветвлением:




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

3.Порядок выполнения работы

3.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

3.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы (или обработчиков событий при использовании визуальных компонентов)
6. Результат тестирования программы.

4.Вопросы к защите

1. Для чего используется условный оператор?
2. Что такое "неполная форма условного оператора"?
3. Как организовать выполнение нескольких операторов при заданном условии?
4. Что такое "вложенное условие"?
5. Каким образом отражается условный оператор в блок-схеме?

5.Индивидуальные задания лабораторной работы

1. Даны две точки $A(x_1,y_1)$ и $B(x_2,y_2)$. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат.
2. Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник, и если да, то будет ли он прямоугольным, остроугольным или тупоугольным.
3. Даны две точки $A(x_1,y_1)$ и $B(x_2,y_2)$. Составить алгоритм, определяющий, которая из точек находится ближе к началу координат. Если точки находятся на одинаковом расстоянии, вывести сообщение об этом.
4. Даны три положительных числа *a, b, c*. Проверить, будут ли они сторонами треугольника. Если да, то определить его тип: равносторонний, равнобедренный, разносторонний.
5. Написать программу, по длинам сторон распознающую среди всех треугольников *ABC* прямоугольные. Если таковых нет, то вычислить величину угла *C*.
6. Написать программу решения уравнения $ax^3 + bx = 0$ для произвольных *a*, *b*.
7. Составить программу, реализующую эпизод применения компьютера в книжном магазине. Если сумма покупки больше 1000 рублей, предоставляется скидка 5%. 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.
8. Определить взаимное расположение точки с координатами (x_0, y_0) и окружности радиуса *R* с центром в точке (x_1, y_1).
9. Если числа *X, Y, Z* попарно различны между собой, то наименьшее из этих трех чисел заменить полусуммой двух других значений. Если же имеются равные числа, вывести сообщение об этом.
10. Число делится на 3 тогда и только тогда, когда сумма его цифр делится на 3. Проверить этот признак на примере вводимого трехзначного числа.
11. Известно, что из четырех чисел a_1, a_2, a_3 и a_4 одно отлично от трех других, равных между собой; присвоить номер этого числа переменной *n*.
12. Вывести на экран номер четверти, которой принадлежит точка с координатами (x; y), при условии, что x и y отличны от 0.

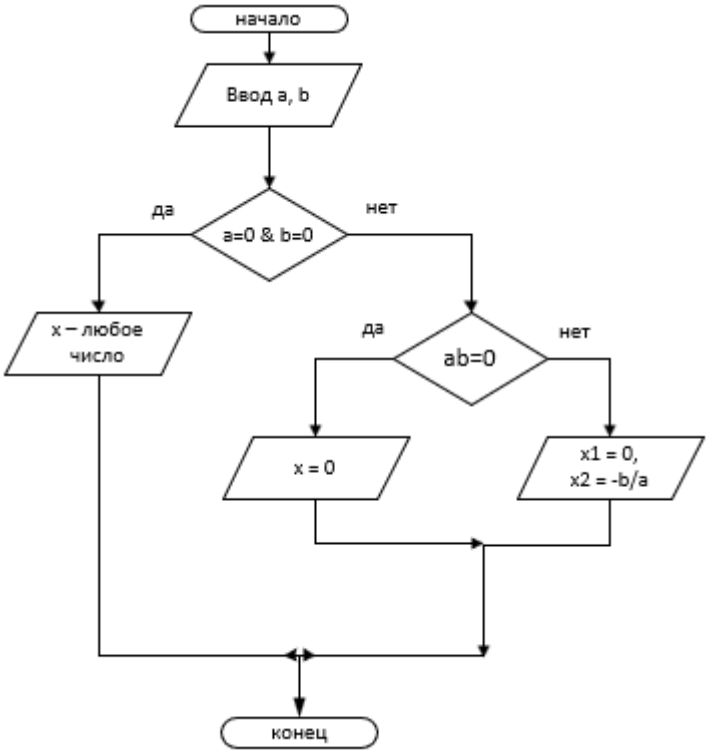
13. Даны целые числа *m, n.* Если числа не равны, то заменить каждое из них одним и тем же числом, равным большему из исходных, а если равны, то заменить числа нулями.
14. На оси *OX* расположены три точки *a, b, c*. Определить, какая из точек *b* или *c* расположена ближе к *a*. Если точки на одинаковом расстоянии, вывести сообщение об этом.
15. Составьте программу, которая определяет сумму только положительных из введенных трех чисел. Если положительных чисел нет, вывести сообщение об этом.

6.Пример выполнения лабораторной работы

6.1.Условие задачи.

Решить квадратное уравнение $ax^2 + b = 0$ для различных значений коэффициентов a и b.

6.2.Блок-схема программы



6.3.Текст программы

```
#include <iostream>
#include<locale.h>  // для переключения на русский язык
using namespace std;
int main()
{
    setlocale(LC_ALL, ".1251");  //  возможность  переключения на русский язык
    double a,b,x;
   // int a, b;

    cout << "\n  Введите коэффициент a = ";
    cin >> a;
    cout << "\n  Введите коэффициент b = ";
    cin >> b;
    if (a == 0 && b == 0)cout << "\n  x - любое число\n";
    else
        if (a * b == 0)cout << "\n  x = 0\n";
        else cout << "\n  x1 = 0;  x2 = " << -b / a<<"\n";
}
```

6.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

1)
```
Введите коэффициент a = 0
Введите коэффициент b = 0
x - любое число
```

2)
```
Введите коэффициент a = 3
Введите коэффициент b = 0
x = 0
```

3)
```
Введите коэффициент a = 2
Введите коэффициент b = 3.5
x1 = 0;  x2 = -1.75
```

ЛАБОРАТОРНАЯ РАБОТА № 7. «ПРОГРАММЫ С ИСПОЛЬЗОВАНИЕМ ОПЕРАТОРА ВЫБОРА»

1. Краткие сведения из теории

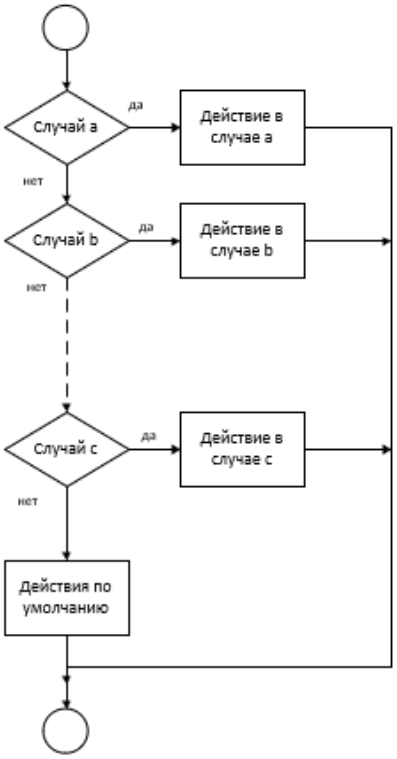
Оператор выбора (переключатель).

Формат оператора выбора:

```
switch (целочисленное выражение)
{   case константа1:список_операторов;
      case константа2: список_операторов;
      ...
      default: список операторов;
}
```

Последняя строка (`default`) может отсутствовать.

Блок-схема оператора выбора имеет вид:



Выполнение оператора происходит в следующем порядке:
1. Вычисляется выражение.
2. Полученное значение последовательно сравнивается с константами, помещенными после служебного слова case; при первом совпадении значений выполняются операторы, стоящие после двоеточия.
3. Если ни с одной из констант совпадения не произошло, то выполнятся операторы после слова `default`.

Для того чтобы «обойти» выполнение операторов на последующих ветвях, нужно принять специальные меры, используя операторы выхода или перехода.

Рассмотрим фрагмент программы, который переводит числовую оценку знаний ученика в ее словесный эквивалент. Согласно вузовской системе: 5 — «отлично», 4 — «хорошо», 3 — «удовлетворительно», 2 — «неудовлетворительно».

Код программы:

```
#include <iostream.h>
void main()
{int ball;
cout<<"\nВведите оценку: "; cin>>ball;
switch (ball)
{ case 2: cout<<"\tЭто неудовлетворительно!\n";
          break;
   case 3: cout<<"\tЭто удовлетворительно!\n";
          break;
   case 4: cout<<"\t Это хорошо!\n"; break;
   case 5: cout<<"\tЭто отлично!\n"; break;
   default: cout<<"\tНет такой оценки!\n";
   }
}
```

Здесь используется еще один новый для нас оператор break — оператор выхода. Его исполнение завершает работу оператора выбора, т.е. происходит «обход» других ветвей. Вот два варианта результатов выполнения этой программы:

```
Введите оценку: 3     Это удовлетворительно!
Введите оценку: 7     Нет такой оценки!
```

Если на всех ветвях убрать оператор `break`, то результат может выглядеть следующим образом:

```
Введите оценку: 3     Это удовлетворительно!
Это хорошо!
Это отлично!
Нет такой оценки!
```

В этом случае выполнились операторы на всех ветвях, начиная с той, которая помечена константой 3.

Возможны задачи, в которых такой порядок выполнения ветвей оператора выбора может оказаться полезным. В следующем фрагменте программы происходит возведение вещественного числа *x* в целую степень *n,* где *n* изменяется в диапазоне от 1 до 5.

```
y=1.0;
switch(n)
{ case 5: y=y*x;
  case 4: y=y*x;
  case 3: y=y*x;
  case 2: y=y*x;
  case 1: y=y*x; cout<<"y="<<y; break;
  default: cout<<"Степень больше 5";
}
```

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.

4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

1.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3. Вопросы к защите

1. В каких случаях целесообразно использование оператора выбора switch?
2. Для чего в операторе выбора применяется оператор break?
3. Какой тип должна иметь переменная-аргумент оператора switch?
4. Для чего используется оператор default?
5. Возможно ли применение оператора выбора без применения операторов break и default?

4. Задание к выполнению

Составить блок-схему алгоритма и написать программу с использованием оператора выбора в соответствии с номером Вашего варианта:

1. Написать программу, которая бы по введенному номеру времени года (1 — зима, 2 — весна, 3 — лето, 4 — осень) выдавала соответствующие этому времени года месяцы, количество дней в каждом из месяцев.
2. Написать программу, которая по вводимому числу от 1 до 5 (номеру курса) выдает соответствующее сообщение. Например, если $k = 1$, «Привет, первокурсник»; если $k = 4$ «Привет, четверокурсник».
3. Составить программу, которая по номеру месяца m определяет количество дней в этом месяце. (Считать год не високосным).
4. Написать программу, которая по номеру дня недели (целому числу от 1 до 7)выдает в качестве результата количество учебных пар в вашей группе в этот день.
5. Имеется пронумерованный список деталей: 1) шуруп, 2) гайка, 3) винт, 4) гвоздь, 5) болт. Составить программу, которая по номеру детали выводит на экран ее название.
6. Написать программу, которая по номеру месяца выдает название следующего за ним месяца (при $m = 1$ получаем февраль, 4 — май и т.д.).
7. Имеется пронумерованный список канцелярских принадлежностей: 1) ручка, 2) карандаш, 3) тетрадь, 4) линейка, 5) скрепка. Составить программу, которая по номеру элемента выводит на экран его название.
8. Написать программу, которая по номеру лабораторной работы выводит ее название.
9. Написать программу, позволяющую по последней цифре числа определить последнюю цифру его квадрата.
10. Для каждой введенной цифры (0 — 9) вывести соответствующее ей название на английском языке (0 — *zero,* 1 — *one, 2 — two,...).*
11. Составить программу, которая по данному числу (1—12) выводит название соответствующего ему месяца.
12. Составить программу, позволяющую по последней цифре данного числа определить последнюю цифру куба этого числа.

13. Написать программу, которая запрашивает у пользователя номер дня недели, затем выводит название дня недели или сообщение об ошибке, если введены неверные данные.
14. Написать программу, которая выводит название учебной дисциплины по номеру пары во вторник.
15. Написать программу, которая по введенному числу от 1 до 12 (номеру месяца) выдает все приходящиеся на этот месяц праздничные дни (например, если введено число 1, то должно получиться 1 января — Новый год, 7 января — Рождество).

5.Пример программы

5.1.Условие задачи.

Составить программу, которая в зависимости от порядкового номера месяца (1, 2, …, 12) выводит на экран время года, к которому относится этот месяц.

5.2.Текст программы

```
#include <iostream>
#include<locale.h>  // для переключения на русский язык
using namespace std;
int main()
{
    setlocale(LC_ALL, ".1251");  //  возможность  переключения на русский язык
    int n;

    cout << "\n  Введите номер месяца n = ";
    cin >> n;
    switch (n) {
    case 1:case 2: case 12: cout << "\n  Зима\n"; break;
    case 3:case 4: case 5: cout << "\n  Весна\n"; break;
    case 6:case 7: case 8: cout << "\n  Лето\n"; break;
    case 9:case 10: case 11: cout << "\n  Осень\n"; break;
    default:  cout << "\n  Нет такого месяца(\n";
    }
}
```

5.3.Тестирование программы

Результат работы программы может выглядеть так:

1)

```
Введите номер месяца n = 5

Весна
```

2)

```
Введите номер месяца n = 15

Нет такого месяца(
```

ЛАБОРАТОРНАЯ РАБОТА №8. ПРИМЕНЕНИЕ ОПЕРАТОРОВ ЦИКЛА В С++ ДЛЯ ВЫЧИСЛЕНИЯ СУММЫ РЯДА

1. Краткие сведения из теории

В С++ существуют три типа операторов цикла: цикл с предусловием, цикл с постусловием и цикл с параметром.

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.

В качестве примера использования оператора цикла рассмотрим программу вычисления факториала целого положительного числа *N*!.

```
// Программа вычисле-
ния факториала
#include <iostream.h>
void main()
{ long int F;
  int i,N;
  cout<<"N="; cin>>N;
  F=i=1;
  while(i<=N) F=F*i++;
 cout<<"\n"<<N<<"!="<<F;
}
```

Обратите внимание на операторы в теле цикла. Конечно, и в программе можно было написать два оператора присваивания, объединив их фигурными скобками. Однако использованный способ записи более лаконичен и более характерен для С++. Этот же самый оператор можно было записать еще короче:`F*=i++.`

При практическом использовании этой программы не следует забывать, что факториал — очень быстро растущая функция, и поэтому при определенных значениях *N* выйдет из диапазона, соответствующего типу `longint`. Задав для переменной *F* тип `unsigned long`, можно сдвинуть эту границу, но этого может оказаться недостаточно. Интересно свойство следующего оператора:

```
while(1);
```

Это бесконечный пустой цикл. Использование в качестве выражения константы 1 приводит к тому, что условие повторения цикла все время остается истинным и работа цикла никогда не заканчивается. Тело в этом цикле представляет собой *пустой оператор.* При исполнении такого оператора программа будет «топтаться на месте».

Рассмотрим еще один пример использования оператора цикла `while`. Рассмотрим задачу итерационного вычисления суммы гармонического ряда: `1+1/2+1/3+...` с заданной точностью ε.

```
// Сумма
//гармонического ряда
#include <iostream.h>
#include <limits.h>
void main()
{int n=1;
 double S=0, eps;
 cout<<"Точность:";
 cin>>eps;
 while(1.0/n>eps &&
 n<INT_MAX)
   S+=1./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=-32768; LONG_MAX=2147483647` и т.д.

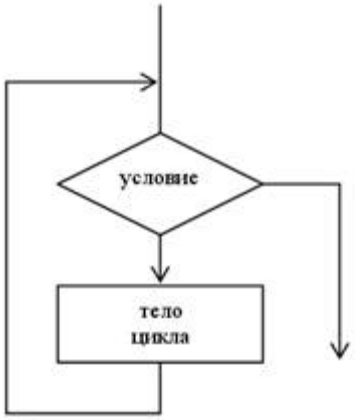
Цикл с постусловием. Формат оператора цикла с постусловием:

do оператор**while** (выражение);

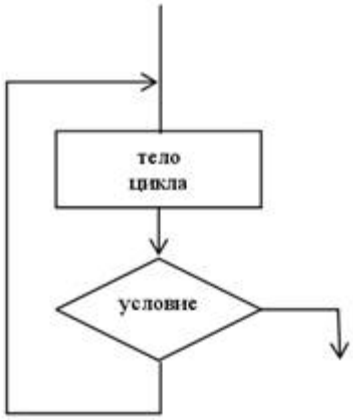
Цикл выполняется до тех пор, пока выражение отлично от нуля, т.е. заключенное в нем условие цикла истинно. Выход из цикла происходит после того, как значение выражения станет ложным, иными словами равным нулю. Таким образом, в отличие от оператора**repeat...until,** используемого в Паскале, где в конце пишется условие выхода из цикла, в операторе**do ...while**в Си в конце пишется условие повторения цикла. В качестве примера рассмотрим программу вычисления *N*!, в которой используется цикл с постусловием.

```
// Программа
вычисления факториала
#include <iostream.h>
void main()
{ long int F;
  int i,N;
  cout<<"N="; cin>>N;
  F=i=1;
  do F*=i++;
while(i<=N);
 cout<<"\n"<<N<<"!="<<F;
}
```

На следующем рисунке представлены блок схемы циклов с предусловием и с постусловием в общем виде:



Цикл с предусловием



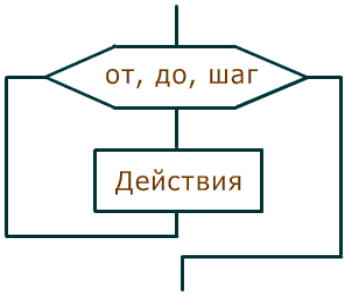
Цикл с постусловием

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 на рис.



С помощью цикла **`for`** нахождение *N*! можно организовать следующим образом:

```
F=1;
for(i=1;i<=N;i++) F=F*i;
```

Используя операцию «запятая», можно в выражение 1 внести инициализацию значений сразу нескольких переменных:

```
for(F=1,i=1;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=1;
i=1;
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++) ;
```

Этот же оператор можно записать в следующей форме:

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

В языке Си оператор**for** является достаточно универсальным средством для организации циклов. С его помощью можно программировать даже итерационные циклы, что невозможно в Паскале. Вот пример вычисления суммы элементов гармонического ряда, превышающих заданную величину ε:

```
for(n=l,S=0;1.0/n>eps && n<INT_MAX;n++) S+=1.0/n;
```

И наконец, эта же самая задача с пустым телом цикла:

```
for(n=l,S=0;1.0/n>eps && n<INT_MAX;S+=1.0/n++);
```

Следующий фрагмент программы на C++ содержит два вложенных цикла**for**. В нем запрограммировано получение на экране таблицы умножения.

```
for(x=2;x<=9;x++)
   for(y=2;y<=9;y++)
        cout<<"\n"<<x<<"*"<<y<<"="<<x*y;
```

На экране будет получен следующий результат:

```
2*2=4
2*3=6
. . .
9*8=72
9*9=81
```

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке VisualC++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Для чего используются операторы цикла?
2. В чём отличие цикла с предусловием и цикла с постусловием?
3. В каких случаях используют цикл по параметру?

4. Какой тип должен иметь параметр функции?
5. Как реализовать выполнение в цикле по параметру нескольких операторов?

4.Задание к выполнению

Составить блок-схему алгоритма и написать программу вычисления и вывода на экран в виде таблицы значений функции, заданной с помощью ряда Тейлора, на интервале от $x_{нач}$ до $x_{кон}$ с шагом dx с точностью ε. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

1. $\ln\frac{x+1}{x-1}=2\sum_{n=0}^{\infty}\frac{1}{(2n+1)x^{2n+1}}=2\left(\frac{1}{x}+\frac{1}{3x^3}+\frac{1}{5x^5}+...\right)\ |x|>1.$

2. $e^{-x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^n}{n!}=1-x+\frac{x^2}{2!}-\frac{x^3}{3!}+\frac{x^4}{4!}-...\ |x|<\infty.$

3. $\ln(x+1)=\sum_{n=0}^{\infty}\frac{(-1)^n x^{n+1}}{n+1}=x-\frac{x^2}{2}+\frac{x^3}{3}-\frac{x^4}{4}-...\ -1<x\le 1.$

4. $\ln\frac{1+x}{1-x}=2\sum_{n=0}^{\infty}\frac{x^{2n+1}}{2n+1}=2\left(x+\frac{x^3}{3}+\frac{x^5}{5}+...\right)\ |x|<1.$

5. $\ln(1-x)=-\sum_{n=1}^{\infty}\frac{x^n}{n}=-\left(x+\frac{x^2}{2}+\frac{x^3}{3}+\frac{x^4}{4}+...\right)\ -1\le x<1.$

6. $arcctgx=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=\frac{\pi}{2}-x+\frac{x^3}{3}-\frac{x^5}{5}+...\ |x|\le 1.$

7. $arctgx=\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+...\ x>1.$

8. $arctgx=\sum_{n=0}^{\infty}\frac{(-1)^{n+1}x^{2n+1}}{2n+1}=x-\frac{x^3}{3}+\frac{x^5}{5}-\frac{x^7}{7}+...\ |x|\le 1.$

9. $arctgx=-\frac{\pi}{2}+\sum_{n=0}^{\infty}\frac{(-1)^{n+1}}{(2n+1)x^{2n+1}}=-\frac{\pi}{2}-\frac{1}{x}+\frac{1}{3x^3}-\frac{1}{5x^5}+...\ x<-1.$

10. $e^{-x^2}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{n!}=1-x^2+\frac{x^4}{2!}-\frac{x^6}{3!}+\frac{x^8}{4!}-...\ |x|<\infty.$

11. $\cos x=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n)!}=1-\frac{x^2}{2!}+\frac{x^4}{4!}-\frac{x^6}{6!}+...\ |x|<\infty.$

12. $\frac{\sin x}{x}=\sum_{n=0}^{\infty}\frac{(-1)^n x^{2n}}{(2n+1)!}=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+...\ |x|<\infty.$

13. $\ln x=2\sum_{n=0}^{\infty}\frac{(x-1)^{2n+1}}{(2n+1)(x+1)^{2n+1}}=2\left(\frac{x-1}{x+1}+\frac{(x-1)^3}{3(x+1)^3}+\frac{(x-1)^5}{5(x+1)^5}+...\right)\ x>0$

14. $\ln x=\sum_{n=0}^{\infty}\frac{(-1)^n(x-1)^{n+1}}{n+1}=(x-1)-\frac{(x-1)^2}{2}+\frac{(x-1)^3}{3}-...\ 0<x\le 2.$

15. $\cos 3x=\sum_{n=0}^{\infty}\frac{(-1)^n(3x)^{2n}}{(2n)!}=1-\frac{(3x)^2}{2!}+\frac{(3x)^4}{4!}-\frac{(3x)^6}{6!}+...\ |x|<\infty$

5.Пример программы

5.1.Условие задачи.

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от $x_{нач}$ до $x_{кон}$ с шагом dx с точностью ε. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots \; |x| < \infty$$

5.2. **Описание переменных**

Xn – начальное значение аргумента x;
Xk – конечное значение аргумента x;
dx – шаг изменения значений аргумента x;
eps – точность вычислений (если слагаемое f становится меньше eps, вычисления прекращаются);
x – текущее значение аргумента;
s – сумма ряда;
f – слагаемое ряда;
i – текущий индекс;
n – количество слагаемых ряда, необходимых для вычисления функции с заданной точностью при некотором значении аргумента x.

5.3. Код программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#include<iomanip>// для форматирования вывода с помощью
манипуляторов
using namespace std;  // использовать пространство имён std
void main(){            // заголовок главной функции
   setlocale(LC_ALL,".1251"); // возможность переключения на
русский язык
float Xn,Xk,dx,eps,x,s,f;
int i,n;
cout<<"\n\nВведите Xn = ";
cin>>Xn;
cout<<"Введите Xk = ";
cin>>Xk;
   cout<<"Введите шаг dx = ";
   cin>>dx;
   cout<<"Введите точность eps = ";
   cin>>eps;
   cout<<"\n        x        S(x)       sin x          n\n"; // Вывод
заголовка таблицы
```

Активация Wir
Чтобы активироват

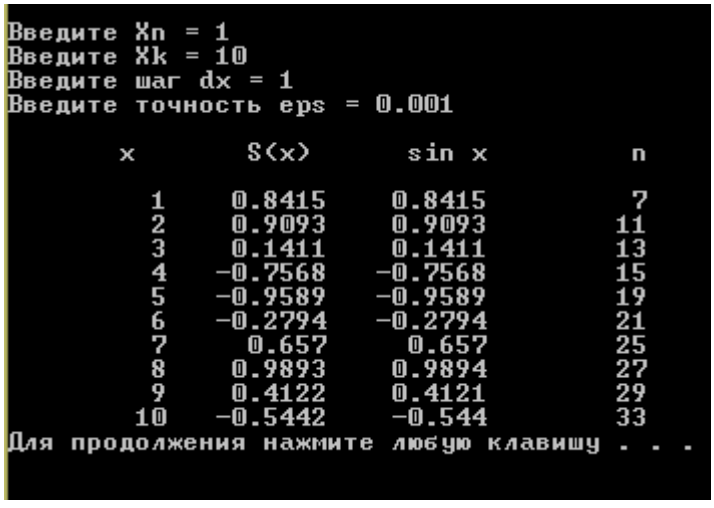
```
n=0;
for(x=Xn; x<=Xk; x+=dx)
{  s=x;
for(f=x,i=3; fabs(f)>eps; i+=2)
       {    f=f*(-x*x/(i-1)/i);    //  f - слагаемое ряда
         s+=f; n=i;              //  s - сумма ряда
}
       cout<<"\n"<<setprecision(4)<<setw(10)<<x //Под переменную
отводится
       <<setprecision(4)<<setw(10)<<s     // 10 позиций, в том числе
       <<setprecision(4)<<setw(10)<<sin(x)    // 4 знака после
запятой
       <<setw(10)<<n;
   }
cout<<"\n";
system("PAUSE");                          //  задержка экрана
}
```

5.4.Тестирование программы

В результате работы программы на экран выводится табличка, снабжённая заголовком.

- Первый столбец таблицы – значения аргумента x, которые мы вводим с клавиатуры: от `Xn до Xk с шагом dx;`
- Второй столбец – значения суммы ряда с заданной точностью;
- Третий – значения стандартной функции, соответствующей заданному ряду;
- Четвёртый столбец – это количество членов ряда, необходимых для вычисления функции с заданной точностью eps.
  Если программа написана верно, то значения второго и третьего столбцов должны быть примерно равны и могут различаться лишь на величину, не превышающую eps.

Результат работы программы может выглядеть так:



ЛАБОРАТОРНАЯ РАБОТА № 9. ПРИМЕНЕНИЕ ОПЕРАТОРОВ ЦИКЛА В С++ ДЛЯ ВЫЧИСЛЕНИЯ ОПРЕДЕЛЁННЫХ ИНТЕГРАЛОВ МЕТОДАМИ ПРЯМОУГОЛЬНИКОВ, ТРАПЕЦИЙ, СИМПСОНА

1. **Краткие сведения из теории**

В тех случаях, когда аналитическое вычисление интеграла затруднено, интеграл можно вычислить приближенно, воспользовавшись одной из следующих формул:

Формула прямоугольников: $$\int_a^b f(x)dx \approx \frac{b-a}{n}\left(y_0 + y_1 + ... + y_{n-1}\right).$$

Формула трапеций: $$\int_a^b f(x)dx \approx \frac{b-a}{n}\left(\frac{y_0 + y_n}{2} + y_1 + y_2 + ... + y_{n-1}\right).$$

Формула Симпсона:

$$\int_a^b f(x)dx \approx \frac{b-a}{6n}\left(y_0 + y_{2n} + 2(y_2 + y_4 + ... + y_{2n-2}) + 4(y_1 + y_3 + ... + y_{2n-1})\right).$$

2. **Порядок выполнения работы**

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Каким образом можно вычислять интегралы, если их аналитическое вычисление затруднено или невозможно?
2. Какие формулы используют для приближённого вычисления интегралов?
3. Провести сравнительный анализ формул приближённого вычисления интегралов. В чем их достоинства и недостатки?
4. С помощью каких операторов реализуются алгоритмы приближённого вычисления определённых интегралов в языке программирования C++?

4.Задание к выполнению

Составить блок-схему алгоритма и написать программу, соответствующую номеру вашего варианта:

1. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 110 $\int_0^3 \sqrt{\frac{x}{6-x}}dx$
2. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 200 $\int_{4\sqrt{2}}^{8} \frac{dx}{x\sqrt{x^2-16}}$.
3. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 55 $\int_1^2 \frac{\sqrt{4-x^2}}{x^2}dx$.
4. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 150 $\int_0^3 \frac{dx}{\sqrt{(16+x^2)^3}}$.
5. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 40 $\int_{\sqrt{3}}^{3} \frac{dx}{x^2\sqrt{x^2+9}}$.
6. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 70 $\int_0^2 \sqrt{(4-x^2)^3}dx$.
7. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 60 $\int_1^3 \frac{\sqrt{x}}{1+x}dx$.
8. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 90 $\int_0^{\frac{\pi}{2}} \sin^3 x \cdot \cos^4 x \ dx$.
9. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 80 $\int_1^8 \frac{dx}{\sqrt{17x+8}}$.
10. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 50 $\int_{-2}^{2} (1-x)\sin(\pi x)dx$.
11. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 75 $\int_{-1}^{0} (2x+3)e^{-x}dx$.
12. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 80 $\int_0^{\pi} (\pi - x)\sin x dx$.
13. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 120 $\int_0^{\frac{\pi}{2}} x^2 \sin x dx$.

14. Вычислить приближенное значение интеграла по формулам трапеций и Симпсона для n = 80 $\int_1^{\sqrt[3]{e}} x^2 \ln x dx$.

15. Вычислить приближенное значение интеграла по формулам прямоугольников и Симпсона для n = 120 $\int_1^2 x \cdot \ln^2 x dx$.

5.Пример программы

Так как наиболее громоздкой и сложной для программирования является формула Симпсона, то именно она рассматривается в разобранном примере. Написать программу приближённого вычисления интеграла по формулам прямоугольников или трапеций нужно самостоятельно.

5.1.Условие задачи.

Вычислить приближенное значение интеграла по формуле Симпсона для n=100 $\int_0^{\pi} x \sin x dx$.

5.2.Описание переменных

a – нижняя граница интеграла;
b – верхняя граница интеграла;
n – количество отрезков разбиения промежутка [a, b];
h – длина элементарного отрезка;
i – текущий индекс;
x – переменная интегрирования (аргумент);
s1 – сумма элементов с чётными номерами;
s2 – сумма элементов с нечётными номерами;
s – итоговая сумма;
res – приближённое значение интеграла.

5.3.Текст программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#define _USE_MATH_DEFINES // для работы с математическими
константами
#include<math.h>
using namespace std;  // использовать пространство имён std
void main(){            // заголовок главной функции
   setlocale(LC_ALL,".1251"); // возможность переключения на русский
   язык
   float a,b,h,x,s1=0,s2=0,s,res;
   int i,n;
   cout<<"Введите нижний предел интегрирования:  ";
   cin>>a;
   b=M_PI;
   cout<<"Введите n:  ";
   cin>>n;
   x=a;
   h=(b-a)/2/n;              //шаг интегрирования
```

```
    for(i=2; i<=2*n-2; i+=2)   //вычисление суммы значений с четными
    номерами
    { s1+=x*sin(x);
          x+=2*h;
     }
     s1*=2;                        //удвоение суммы значений с четными
    номерами
       x=a;
    for(i=1; i<=2*n-1; i+=2)  //вычисление суммы значений с нечетными
    номерами
    { s2+=x*sin(x);
          x+=2*h;
     }
     s2*=4;
     s=s1+s2+a*sin(a)+b*sin(b); //вычисление итоговой суммы
    res=(b-a)*s/(6*n);          //приближенное вычисление интеграла
    cout<<"Integral = "<<res<<endl;
    system("PAUSE");                        //  задержка экрана
}
```

Активация Windov
Чтобы активировать Wi

5.4.Тестирование программы

В каждом индивидуальном задании указано два способа, которыми должен быть вычислен интеграл (например, по формуле Симпсона и по формуле прямоугольников). Результаты вычислений обоими методами должны быть близки по значению. Кроме того, для проверки полученных результатов можно вычислить интеграл аналитически.
Для рассматриваемого интеграла:

$$\int_0^{\pi} x\sin x dx = -x\cos x\Big|_0^{\pi} + \int_0^{\pi}\cos x dx = -x\cos x\Big|_0^{\pi} + \sin x\Big|_0^{\pi} = (-\pi\cos\pi + 0) = \pi$$

$u = x \;\; du = dx$

$dv = \sin x dx \;\; v = -\cos x$.

Запуская программу на выполнение, убеждаемся, что в результате действительно получаем приближенное значение числа π.

ЛАБОРАТОРНАЯ РАБОТА №10. РАБОТА С ЭЛЕМЕНТАМИ ОДНОМЕРНЫХ МАССИВОВ.

1. Краткие сведения из теории

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.
Рассмотрим несколько примеров программ обработки одномерных массивов.
Пример 1. Ввод с клавиатуры и вывод на экран одномерного массива.

```
//Ввод и вывод массива
#include <iostream>
using namespace std;
void 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]<<"    ";
}
```

Пример2. Ввод вещественного массива и вычисление среднего значения.

```
#include <iostream>
using namespace std;
void main ()
{ const n=10;
int i;
double a[n],sa;
     for(i=0; i<n;i++)
   {  cout«"a["«i«"]=";
          cin>>a[i];
           }
for(i=0; i<n; i++) sa+=a[i];
     sa/=n;
     cout<<"\Среднее арифметическое массива=",sa;
}
```

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

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения (общая часть, если таковая имеется, и индивидуальное задание).
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Что такое одномерный массив?
2. Как задать одномерный массив в C++?
3. С какого числа начинается нумерация массивов в C++?
4. В каких случаях используются массивы?
5. Могут ли быть элементы одного массива разных типов?
6. Что такое размерность массива?

4.Задания лабораторной работы № 10

Составить блок-схему алгоритма и написать программу на языке C++ в соответствии с номером вашего варианта:

1. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   а) сумму отрицательных элементов массива;

b) произведение элементов массива, расположенных между максимальным и минимальным элементами.

2. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   a) сумму положительных элементов массива;
   b) произведение элементов массива, расположенных после максимального по модулю элемента;
3. В одномерном массиве, состоящем из n целых элементов, вычислить:
   a) произведение элементов массива с четными номерами;
   b) сумму элементов массива, расположенных между первым и последним нулевыми элементами;
4. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   a) сумму элементов массива с нечетными номерами;
   b) сумму элементов массива, расположенных между первым и последним отрицательными элементами.
5. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   a) максимальный элемент массива;
   b) сумму элементов массива, расположенных до последнего положительного элемента.
6. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   a) минимальный элемент массива;
   b) сумму элементов массива, расположенных между первым и последним положительными элементами.
7. В одномерном массиве, состоящем из n целых элементов, вычислить:
   a) номер максимального элемента массива;
   b) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.
8. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   a) номер минимального элемента массива;
   b) сумму элементов массива, расположенных между первым и вторым отрицательными элементами.
9. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
   a) максимальный по модулю элемент массива;
   b) сумму элементов массива, расположенных между первым и вторым положительными элементами.
10. В одномерном массиве, состоящем из n целых элементов, вычислить:
    a) минимальный по модулю элемент массива;
    b) сумму модулей элементов массива, расположенных после первого элемента, равного нулю.
11. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
    a) номер минимального по модулю элемента массива;
    b) сумму модулей элементов массива, расположенных после первого отрицательного элемента.
12. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
    a) номер максимального по модулю элемента массива;
    b) сумму элементов массива, расположенных после первого положительного элемента.
13. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
    a) количество отрицательных элементов массива;
    b) сумму элементов массива, расположенных после максимального элемента.
14. В одномерном массиве, состоящем из n вещественных элементов, вычислить:
    a) количество элементов массива, равных 0;
    b) сумму элементов массива, расположенных после минимального элемента.
15. В одномерном массиве, состоящем из n целочисленных элементов, вычислить:
    a) сумму элементов массива, делящихся нацело на 3;

b) количество элементов массива, расположенных после минимального элемента.

5.Пример выполнения лабораторной работы

5.1. Условие задачи

В одномерном массиве, состоящем из n вещественных элементов, вычислить:
a) количество положительных элементов массива;
b) сумму модулей элементов массива, расположенных после последнего элемента, равного нулю;

5.2. Используемые переменные

`k` – количество положительных чисел;
`n` – общее количество чисел в массиве;
`s` – сумма элементов массива после последнего нулевого элемента;
`x[i]` – элементы массива;
`k1` – номер последнего нуля;
`i` – индекс, используемый в циклах.

5.3.Текст программы

```
#include<iostream>  //  библиотека ввода-вывода
#include <ctime>    // для работы со случайными числами
using namespace std;  // использовать пространство имён std

int main() {           // заголовок главной функции
    setlocale(0, "");    //  возможность  переключения на русский язык
    int i, k = 0, k1, n;
    srand(time(0));    // инициализация датчика случайных чисел
    float s = 0, x[50];
    cout << "\nВведите  n = ";
    cin >> n;           //  размерность массива
    k1 = n;
    for (i = 0; i < n; i++)
    {
        x[i] = rand() % 8 - 3;  //  заполнение массива случайными числами
        cout << "  " << x[i];  //  вывод массива на экран
    }
    for (i = 0; i < n; i++)
    {
        if (x[i] > 0) k++;   // количество положительных элементов
        if (x[i] == 0) k1 = i;  // номер последнего нуля
    }
    cout << "\nКоличество положительных элементов массива k = " << k;

    for (i = k1+1; i < n; i++)
         s += x[i];
    cout << "\nСумма после последнего нуля s = " << s<<endl;
}
```

5.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

Тест 1.

```
Введите  n = 7
  -3  -3  1  -3  1  2  -2
Количество положительных элементов массива k = 3
Сумма после последнего нуля s = 0
```

Тест 2.

```
Введите  n = 8
  0  -2  4  0  -3  -1  -2  -1
Количество положительных элементов массива k = 1
Сумма после последнего нуля s = -7
```

ЛАБОРАТОРНАЯ РАБОТА №11. СОРТИРОВКИ В ОДНОМЕРНЫХ МАССИВАХ

1.Краткие сведения из теории

Для программных продуктов часто используемыми являются алгоритмы сортировки данных — упорядочения информации по некоторому признаку. От эффективности, прежде всего скорости, их выполнения во многом зависит эффективность работы всей программы.

Как правило, сортируемые данные располагаются в массивах. В простейшем случае это числовые массивы. Существует множество различных способов сортировки. Приведем пример сортировки массива по возрастанию "методом пузырька".

```
#include<iostream>  //  библиотека ввода-вывода
#include <ctime>    // для работы со случайными числами

using namespace std;  // использовать пространство имён std
int main() {           // заголовок главной функции
    setlocale(0, "");    //  возможность  переключения на русский язык
    int i, j, n;
    srand(time(0));    // инициализация датчика случайных чисел
    float x[50], a;
    cout << "\n Введите  n = ";
    cin >> n;           //  размерность массива
    for (i = 0; i < n; i++)
    {
        x[i] = rand() % 51-25;  //  заполнение массива случайными числами
        cout << "  " << x[i];  //  вывод массива на экран
    }
    for (i = 0; i < n-1; i++)   //  сортировка
        for (j = 0; j < n-1-i; j++)
            if (x[j] > x[j + 1]) {
                a = x[j];
                x[j] = x[j + 1];
                x[j + 1] = a;
            }
    cout <<"\n Преобразованный массив\n";
    for (i = 0; i < n; i++)
        cout<< "  " << x[i];
    cout << endl;

}
```

Результат может выглядеть так:

```
Введите  n = 7
 7  -15  21  -6  -2  11  -18
Преобразованный массив
 -18  -15  -6  -2  7  11  21
```

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения (общая часть, если таковая имеется, и индивидуальное задание).
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Что такое одномерный массив?
2. Как задать одномерный массив в C++?
3. С какого числа начинается нумерация массивов в C++?
4. Для чего используются сортировки?

4.Задания лабораторной работы № 11

Составить блок-схему алгоритма и написать программу на языке C++ в соответствии с номером вашего варианта:

Список индивидуальных заданий лабораторной работы № 11

1. Преобразовать одномерный массив из n целых элементов таким образом, чтобы сначала располагались все чётные элементы, а потом - все нечётные (элементы, равные 0, считать чётными).
2. Преобразовать одномерный массив из n целых элементов таким образом, чтобы сначала располагались все положительные элементы, а потом - все отрицательные (элементы, равные 0, считать положительными).
3. Дан одномерный массив, состоящий из n вещественных элементов. Сжать массив, удалив из него все элементы, модуль которых не превышает 1. Освободившиеся в конце массива элементы заполнить нулями.
4. Сжать одномерный массив, состоящий из n вещественных элементов, удалив из него все элементы, модуль которых находится в интервале [a; b]. Освободившиеся в конце массива элементы заполнить нулями.
5. Преобразовать одномерный массив, состоящий из nвещественных элементов, таким образом, чтобы сначала располагались все элементы, равные 0, а потом – все остальные.
6. Преобразовать одномерный массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине – элементы, стоявшие в четных позициях.
7. Преобразовать одномерный массив, состоящий из nвещественных элементов таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом все остальные.
8. Сжать одномерный массив, состоящий из n целых элементов, удалив из него все чётные элементы. Освободившиеся в конце массива элементы заполнить единицами.
9. Преобразовать одномерный массив таким образом, чтобы элементы, равные 0, располагались после всех остальных.
10. Преобразовать одномерный массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в четных позициях, а во второй половине – элементы, стоявшие в нечетных позициях.

11. Сжать одномерный массив, удалив из него все элементы, величина которых находится в интервале [a; b]. Освободившиеся в конце массива элементы заполнить нулями.
12. Преобразовать одномерный массив из n целых элементов таким образом, чтобы сначала располагались элементы, делящиеся на 3 нацело, а потом – все остальные.
13. Упорядочить элементы одномерного массива по убыванию модулей элементов.
14. Сжать одномерный массив, состоящий из n целых элементов, удалив из него все элементы, делящиеся нацело на 3. Освободившиеся в конце массива элементы заполнить нулями.
15. Упорядочить элементы одномерного массива по возрастанию модулей элементов.

5.Пример выполнения лабораторной работы

5.1. Условие задачи

Задан одномерный массив вещественных чисел. Преобразовать массив таким образом, чтобы сначала располагались все элементы, меньшие 2, а потом - все остальные.

5.2. Используемые переменные

`n` – общее количество чисел в массиве;
`x[i]` – элементы массива;
y[i] – вспомогательный массив;
`i, j` – индексы, используемые в циклах.

5.3.Текст программы

```
#include<iostream>  //  библиотека ввода-вывода
#include <ctime>    // для работы со случайными числами

using namespace std;  // использовать пространство имён std
int main() {           // заголовок главной функции
    setlocale(LC_ALL, ".1251");    //  возможность  переключения на русский язык
    int i, j = 0, n;
    srand(time(0));    // инициализация датчика случайных чисел
    float x[50], y[50];
    cout << "\n Введите  n = ";
    cin >> n;           //  размерность массива
    for (i = 0; i < n; i++)
    {
        x[i] = (rand() % 20-4)/2.;  //  заполнение массива случайными числами
        cout << "   " << x[i];  //  вывод массива на экран
    }
    for (i = 0; i < n; i++)
        if (x[i] < 2) {
            y[j] = x[i];
            j++;
        }
    for (i = 0; i < n; i++)
        if (x[i] >= 2) {
            y[j] = x[i];
            j++;
        }
    cout << "\n Отсортированный массив\n";
    for (i = 0; i < n; i++)
        cout << "   " << y[i];  //  вывод массива на экран
}
```

5.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

```
Введите  n = 11
 -0.5  -2  2.5  6  4  7  0  0.5  7.5  4  -0.5
Отсортированный массив
 -0.5  -2  0  0.5  -0.5  2.5  6  4  7  7.5  4
```

ЛАБОРАТОРНАЯ РАБОТА № 12. ПРОГРАММИРОВАНИЕ ВЛОЖЕННЫХ ЦИКЛОВ НА ПРИМЕРЕ РЕБУСОВ

1. Краткие сведения из теории

Цикл for

Цикл for имеет следующее формальное определение:

```
for (выражение_1; выражение_2; выражение_3)
{
    // тело цикла
}
```

выражение_1 - выполняется один раз при начале выполнения цикла и представляет установку начальных условий, как правило, это инициализация счетчиков - специальных переменных, которые используются для контроля за циклом.

выражение_2 - представляет условие, при соблюдении которого выполняется цикл. Как правило, в качестве условия используется операция сравнения, и если она возвращает ненулевое значение (то есть условие истинно), то выполняется тело цикла, а затем вычисляется *выражение_3*.

выражение_3 - задает изменение параметров цикла, нередко здесь происходит увеличение счетчиков цикла на единицу.

Можно определять вложенные циклы. Например, выведем таблицу умножения:

```
#include <iostream>
 using namespace std;
int main()
{
    for (int i=1; i < 10; i++)
    {
        for(int j = 1; j < 10; j++)
        {
            cout << i * j << "\t";
        }
        cout << endl;
    }

    return 0;
}
```

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.
7.
8.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.

5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Для чего используются операторы цикла?
2. В каких случаях используют цикл по параметру?
3. Какой тип должен иметь параметр функции?
4. Как реализовать выполнение в цикле по параметру нескольких операторов?

4.Задание к выполнению

Составить алгоритм и написать программу решения ребуса в соответствии с индивидуальным заданием:

1. Составить алгоритм решения ребуса **РАДАР** = **(Р** + **А** + **Д)**^{4} (различные буквы обозначают различные цифры, старшая — не 0).

2. Составить алгоритм решения ребуса **ДРУГ — ГУРД** = 2727 (различные буквы обозначают различные цифры, старшая — не 0).

3. Заменить буквы цифрами так, чтобы соотношение оказалось верным (одинаковым буквам соответствуют одинаковые цифры, разным — разные, старшая цифра не 0):

ХРУСТ·ГРОХОТ = РРРРРРРРРРР.

4. Составить алгоритм решения ребуса **вагон + вагон = состав** (различные буквы обозначают различные цифры, старшая – не 0).

5. Найти все решения ребуса **море + шторм = авария** (различные буквы обозначают различные цифры, старшая – не 0).

6. Составить алгоритм решения ребуса **барс + рысь = кошки** (различные буквы обозначают различные цифры, старшая – не 0).

7. Составить алгоритм решения ребуса **ситец + ситец = платье** (различные буквы обозначают различные цифры, старшая – не 0).

8. Составить алгоритм решения ребуса **вода + песок = оазис** (различные буквы обозначают различные цифры, старшая – не 0).

9. Составить алгоритм решения ребуса **зерно + зерно = колос** (различные буквы обозначают различные цифры, старшая – не 0).

10. Составить алгоритм решения ребуса **карп + арп + рп + п = рыба** (различные буквы обозначают различные цифры, старшая – не 0).

11. Составить алгоритм решения ребуса **курск + горск = города** (различные буквы обозначают различные цифры, старшая – не 0).

12. Составить алгоритм решения ребуса

FORTY + TEN + TEN = SIXTY (различные буквы обозначают различные цифры, старшая – не 0).

13. Составить алгоритм решения ребуса

сосна + сосна + сосна + сосна + сосна =тайга

(различные буквы обозначают различные цифры, старшая – не 0).

14. Составить алгоритм решения ребуса

два
х
сто
———
двести

(различные буквы обозначают различные цифры, старшая – не 0).

15. Составить алгоритм решения ребуса **МУХА + МУХА + МУХА = СЛОН** (различные буквы обозначают различные цифры, старшая — не 0).

16. Составить алгоритм решения ребуса **КТО + КОТ = ТОК** (различные буквы обозначают различные цифры, старшая — не 0).

5.Пример программы

5.1.Условие задачи.

Составить алгоритм решения ребуса **буква + буква + буква = слово**
(различные буквы обозначают различные цифры, старшая – не 0).

5.2.Используемые переменные

buk - слагаемое
slo - сумма
a[8]- массив цифр
i,j - индексы
r - вспомогательная переменная для проверки различности цифр.

5.3.Текст программы

```
#include<iostream>
using namespace std;
void main()
{
    int i, j, r;
    int buk, slo, a[8];

    for (a[0] = 1; a[0] <= 3; a[0]++)
     for (a[1] = 0; a[1] <= 9; a[1]++)
      for (a[2] = 0; a[2] <= 9; a[2]++)
       for (a[3] = 0; a[3] <= 9; a[3]++)
         for (a[4] = 0; a[4] <= 9; a[4]++)
          for (a[5] = 1; a[5] <= 9; a[5]++)
           for (a[6] = 0; a[6] <= 9; a[6]++)
            for (a[7] = 0; a[7] <= 9; a[7]++)
            {
                buk = a[0] * 10000 + a[1] * 1000 + a[2] * 100 + a[3] * 10 + a[4];
                slo = a[5] * 10000 + a[6] * 1000 + a[7] * 100 + a[3] * 10 + a[7];

                if (buk * 3 == slo)
                {
                    r = 0;                    //Проверка того, что
                    for (i = 0; i < 8; i++)     //разным буквам соответствуют
                        for (j = i + 1; j < 8; j++)  //разные цифры
                            if (a[i] == a[j]) r++;
                        if (r == 0) cout << buk << " + "
                                         << buk << " + "
                                         << buk << " = "
                                         << slo << endl;
                }
            }
}
```

Результат работы программы:

```
23047 + 23047 + 23047 = 69141
26451 + 26451 + 26451 = 79353
32047 + 32047 + 32047 = 96141
```

ЛАБОРАТОРНАЯ РАБОТА №13. ДИНАМИЧЕСКИЕ МАССИВЫ

1. Краткие сведения из теории

Указатели и динамические массивы.

Если до начала работы программы неизвестно количество элементов массива, можно использовать динамические массивы.

Для выделения памяти под динамический массив используется оператор **new**, после которого в квадратных скобках указывается, сколько массив будет содержать объектов:

```
int *numbers = new int[4]; // динамический массив из 4 чисел
```

Причем в этом случае оператор **new** также возвращает указатель на объект типа `int` – первый элемент в созданном массиве.

В данном случае определяется массив из четырех элементов типа `int`, но каждый из них имеет неопределенное значение. Однако мы также можем инициализировать массив значениями:

```
int *n1 = new int[4];  // каждый элемент имеет неопределенное значение
int *n2 = new int[4](); // каждый элемент имеет значение по умолчанию - 0
int *n3 = new int[4]{ 1, 2, 3, 4 }; // массив состоит из чисел 1, 2, 3, 4
```

В последнем случае при инициализации массива конкретными значениями следует учитывать, что если значений в фигурных скобках больше чем длина массива, то оператор new потерпит неудачу и не сможет создать массив. Если переданных значений, наоборот, меньше, то элементы, для которых не предоставлены значения, инициализируются значением по умолчанию.

После создания динамического массива мы сможем с ним работать по полученному указателю, получать и изменять его элементы:

```
int n = 5;  // размер массива
int *p = new int[n]{ 1, 2, 3, 4, 5 };
for (int *q = p; q != p + n; q++)
{
    std::cout << *q << "\t";
}
```

Для удаления динамического массива и освобождения его памяти применяется специальная форма оператора **delete:**

```
delete [] указатель_на_динамический_массив;
```

Например:

```
#include <iostream>
int main()
{
    int n = 5;  // размер массива
    int *p = new int[n]{ 1, 2, 3, 4, 5 }; // массив состоит из чисел
1, 2, 3, 4
    for (int *q = p; q != p + n; q++)
    {
        std::cout << *q << "\t";
    }
    std::cout << std::endl;
    delete [] p;
    return 0;
}
```

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1.Номер работы и её название.

2.Задание для выполнения (общая часть, если таковая имеется, и индивидуальное задание).

3.Код программы на языке Visual C++.

4.Описание входных и выходных данных.

5.Блок-схема программы.

6.Результат тестирования программы.

3.Вопросы к защите

1. Что такое динамический массив?
2. Как задать динамический массив в C++?
3. С какого числа начинается нумерация массивов в C++?
4. В каких случаях используются динамические массивы?
5. Что такое размерность массива?

4.Задания лабораторной работы № 13

Составить блок-схему алгоритма и написать программу на языке C++ в соответствии с номером вашего варианта:

1. Вычислить произведение отрицательных элементов P_1 последовательности действительных чисел $a_1, a_2, ..., a_n$ и произведение положительных элементов P_2. Сравнить модуль P_2 с модулем P_1, указать, какое из произведений по модулю больше.
2. Дан массив действительных чисел. Среди них есть равные. Найти его первый максимальный элемент и заменить его нулем.
3. Дана последовательность целых чисел $a_1, a_2, ..., a_n$. Образовать новую последовательность, выбросив из исходной те члены, которые равны $\min(a_1, a_2, ..., a_n,)$.
4. У прилавка магазина выстроилась очередь из n покупателей. Время обслуживания i-го покупателя равно t_i ($i = 1, ..., n$). Определить время C_i пребывания i-го покупателя в очереди.
5. Дана последовательность целых чисел $a_1, a_2, ..., a_n$. Указать пары чисел a_i, a_j, таких, что $a_i + a_j = m$.
6. Дана последовательность целых чисел $a_1, a_2, ..., a_n$. Наименьший член этой последовательности заменить целой частью среднего арифметического всех членов, остальные члены оставить без изменения. Если в последовательности несколько наименьших членов, то заменить последний по порядку.
7. Даны две последовательности целых чисел $a_1, a_2, ..., a_n$ и $b_1, b_2, ..., b_n$. Преобразовать последовательность $b_1, b_2, ..., b_n$ по следующему правилу: если $a_i \le 0$, то b_i увеличить в 10 раз, в противном случае b_i заменить нулем ($i = 1, 2, ..., n$).
8. Дана последовательность действительных чисел $a_1, a_2, ..., a_n$. Требуется домножить все члены последовательности $a_1, a_2, ..., a_n$ на квадрат ее наименьшего члена, если $a_k \ge 0$, и на

квадрат ее наибольшего члена, если $a_k < 0$.

9. Даны координаты n точек на плоскости: (X_1, Y_1),..., (X_n, Y_n). Найти номера пары точек, расстояние между которыми наибольшее (считать, что такая пара единственная).
10. Вывести информацию о наименее удаленном от среднего арифметического члена последовательности вещественных чисел.
11. В одномерном массиве с четным количеством элементов ($2N$) находятся координаты N точек плоскости. Они располагаются в следующем порядке: x_1, y_1, x_2, y_2, x_3, y_3 и т.д. Определить минимальный радиус окружности с центром в начале координат, которая содержит все точки.
12. Дан массив из n четырехзначных натуральных чисел. Вывести на экран только те, у которых сумма первых двух цифр равна сумме двух последних.
13. Даны две последовательности целых чисел a_1, a_2,..., a_n и b_1, b_2,..., b_n. Все члены последовательностей — различные числа. Найти, сколько членов первой последовательности совпадает с членами второй последовательности.
14. Напишите программу, входными данными которой является возраст n человек. Программа подсчитывает количество людей, возраст которых находится в интервале 10 лет, а именно:
    0-9 лет;
    10-19 лет;
    20—29 лет и т. д.
    Напечатать результаты расчетов в удобочитаемой форме.
15. Вывести информацию о наиболее удаленном от среднего арифметического члена последовательности вещественных чисел.

5.Пример выполнения лабораторной работы

5.1. Условие задачи

Дан целочисленный массив $A[n]$, среди элементов есть одинаковые. Создать массив из различных элементов $A[n]$.

5.2. Используемые переменные

`n` – размерность исходного массива;
`x[i]` – элементы исходного массива;
`l` – размерность полученного массива;
`y[i]` – элементы полученного массива;
ko1 - вспомогательная переменная, используемая при сравнении элементов;
i, j – индексы, используемые в циклах.

5.3.Текст программы

```
#include <iostream>
#include<locale.h>
#include <ctime>    // для работы со случайными числами
using namespace std;
int main()
{
    srand(time(0));
    setlocale(LC_ALL, ".1251");
    int n,l=0,kol;
    cout << "n="; cin >> n;
    int* x = new int[n];
    int* y = new int[l];
    for (int i=0;i<n;i++)
    {
        x[i] = rand() % 8-3;
        cout << "  " << x[i];
    }
```

```
    for (int i = 0; i < n; i++)
    {
        kol = 0;
        for (int j = 0; j < n; j++)
            if (x[i] != x[j]) kol++;
        if (kol == n - 1)
        {   y[l] = x[i];
            l++;
        }
    }
    cout << endl;
    for (int i = 0; i < l; i++)
        cout << "  " << y[i];
    if (l == 0) cout << "\n Все числа повторяются\n";
    delete[] x;
    delete[] y;
}
```

5.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:

```
 n = 7
1  2  2  -1  2  -3  1
-1  -3

 n = 5
1  1  1  0  0

Все числа повторяются
```

ЛАБОРАТОРНАЯ РАБОТА №14. .ЗАДАЧИ НА ФОРМИРОВАНИЕ ДВУМЕРНЫХ МАССИВОВ
ЛАБОРАТОРНАЯ РАБОТА №15. ОБРАБОТКА ДВУМЕРНЫХ МАССИВОВ

1. **Краткие сведения из теории**

Многомерные массивы

Двумерный массив трактуется как одномерный массив, элементами которого являются массивы с указанным в описании типом элементов. Например, оператор

```
Float R[5][10];
```

объявляет массив из пяти элементов, каждый из которых есть массив из десяти вещественных чисел. Отдельные величины этого массива обозначаются именами с двумя индексами: `R[0][0], R[0][1], ...,R[4][9]`. Объединять индексы в одну пару скобок нельзя, т. е. запись `R[2,3]` ошибочна. Пример описания трехмерного массива:

```
double X[3][7][20];
```

Порядок расположения элементов многомерного массива в памяти такой, что прежде всего меняется последний индекс, затем предпоследний и т.д., и лишь один раз пробегает свои значения первый индекс.
При описании многомерных массивов их также можно инициализировать. Делать это удобно так:

```
int M[3][3]={ 11,12,13,
              21,22,23,
              31,32,33};
```

Рассмотрим примеры программ обработки матриц — числовых двумерных массивов.

Пример 1. Вычисление и вывод на экран таблицы умножения в форме матрицы Пифагора.

```
#include <iostream>
using namespace std;
void main ()
{ int i,j, a[10][10];
      for(i=1; i<=9; i++)
        {for(j=1; j<=9; j++)
           {  a[i][j]=i*j;
              cout<<a[i][j]<<"    ";
           }
        cout<<endl;
    }
}
```

Пример 2. Заполнение матрицы случайными числами в диапазоне от 0 до 99 и поиск в ней максимального значения.

```
#include <iostream>
#include <iomanip.h>
#include <ctime>
#define n 5
using namespace std;
void main ()
{   int i, j , ImaxA, JmaxA, A[n][n];
srand(time(0));
//Установка датчика случайных чисел
for(i=0;  i< n; i++)
{ for(j=0; j<n; j++)
{ A[i][j]=rand()%100;
cout<<setw(6)<<A[i][j];
     }
cout<<endl;
}
ImaxA=JmaxA=0;
for(i=0; i<n; i++)
 {  for(j=0; j<n; j++)
      if (A[i][j]>A[ImaxA][JmaxA])
        {  ImaxA=i;  JmaxA=j;   }
 }
cout<<"\nМаксимальное_значение="<<A[ImaxA][JmaxA]<<endl;
system("pause");
}
```

Активация \
Чтобы активир
раздел "Парам

В результате тестирования этой программы получен следующий результат:

```
46    23    57    35    18
8     48    68     4    70
56    98    16    71    40
70    84    66    67    11
20    44    37    57    38

Максимальное значение: A[2][1]=98
```

Указатели и массивы.

Имя массива трактуется как указатель-константа на массив.

Пусть, например, в программе объявлен массив:

```
int X[10];
```

В таком случае X является указателем на нулевой элемент массива в памяти компьютера. В связи с этим истинным является отношение

```
X==&X[0]
```

Отсюда следует, что для доступа к элементам массива кроме индексированных имен можно использовать разадресованные указатели по принципу:

```
имя [индекс]   тождественно * (имя + индекс)
```

Например, для описанного выше массива x взаимозаменяемы следующие обозначения элементов:

```
X[5], или *(X+5),или *(5+X).
```

Напоминаем, что для указателей работают свои правила сложения. Поскольку `X` — указатель на величину целого типа, то `X+5` увеличивает значение адреса на 10.

В языке C++ символ [*играет роль знака операции сложения адреса массива с индексом*

элемента массива.

Из сказанного должно быть понятно, почему индекс первого элемента массива всегда нуль. Его адрес должен совпадать с адресом массива:

```
X[0]==*(X+0)
```

Поскольку имя массива является указателем-константой, то его нельзя изменять в программе, т. е. ему нельзя ничего присваивать. Например, если описаны два одинаковых по структуре массива

```
int X[10],Y[10];
```

то оператор присваивания `X=Y` будет ошибочным. Такое возможно в Паскале, но недопустимо в С++. Пересылать значения одного массива в другой можно только поэлементно.

Теперь рассмотрим двумерные массивы. Пусть в программе присутствует описание:

```
int P[5][10];
```

Это матрица из пяти строк и десяти чисел в каждой строке. Двумерный массив расположен в памяти в последовательности по строкам. По-прежнему `P` является указателем-константой на массив, т. е. на элемент `P[0][0]`. Индексированное имя `P[i]` обозначает *i*-ю строку. Ему тождественно следующее обозначение в форме разадресованного указателя:

```
*(P+i*10)
```

Обращение к элементу массива `P[2][4]` можно заменить на `*(P+2*10+4)`. В общем случае эквивалентны обозначения:

```
P[i] [j]    и    *(P+i*10+j)
```

Здесь дважды работает операция «квадратная скобка». Последнее выражение можно записать иначе, без явного указания на длину строки матрицы `P`:

```
*(*(P+i)+j).
```

Очевидно, что по индукции для ссылки на элемент трехмерного массива`A[i][j][k]` справедливо выражение `*(*(*(A+i)+j)+k)` и т.д.

2. Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2.Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения (общая часть, если таковая имеется, и индивидуальное задание).
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

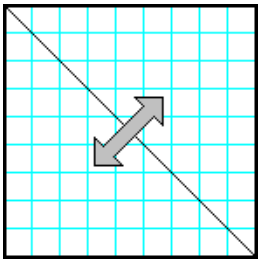
3.Вопросы к защите

1. Что такое двумерный массив?
2. Как задать двумерный массив в С++?
3. Что такое размерность двумерного массива?
4. Какой индекс массива меняется в первую очередь?
5. Для чего можно использовать многомерные массивы?

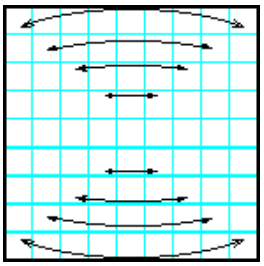
4.Задания лабораторной работы № 14

Составить блок-схему алгоритма и написать программу на языке С++ в соответствии с номером вашего варианта:

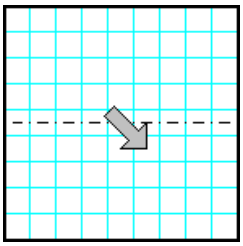
1. Заполнить матрицу случайными числами. Развернуть матрицу на 90º по часовой стрелке.
2. Заполнить матрицу случайными числами. Отобразить матрицу симметрично относительно главной диагонали



3. Заполнить матрицу случайными числами. На главной диагонали разместить суммы элементов, которые лежат на соответствующих строках.
4. Заполнить матрицу случайными числами. Отобразить главную и побочную диагонали симметрично относительно вертикальной оси.

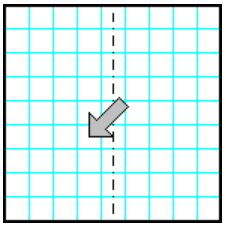


5. Заполнить матрицу случайными числами. Отобразить верхнюю половину матрицы на нижнюю зеркально симметрично относительно горизонтальной оси.

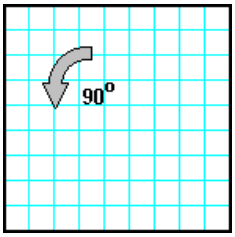


6. Заполнить матрицу случайными числами. На главной диагонали разместить суммы элементов соответствующих столбцов.

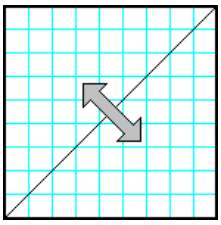
7. Заполнить матрицу случайными числами. Отобразить правую половину матрицы на левую зеркально симметрично относительно вертикальной оси.



8. Заполнить матрицу случайными числами. Развернуть матрицу на 90º против часовой стрелки.



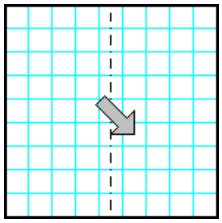
9. Заполнить матрицу случайными числами. Отобразить матрицу симметрично относительно побочной диагонали



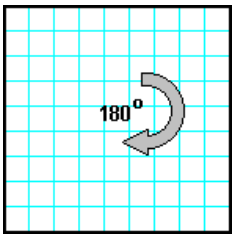
10.Заполнить матрицу случайными числами. На побочной диагонали разместить суммы элементов, расположенные на соответствующих строках.

11.Заполнить матрицу случайными числами. Отобразить главную и побочную диагонали симметрично относительно горизонтальной оси.

12.Заполнить матрицу случайными числами. Отобразить левую половину матрицы на правую зеркально симметрично относительно вертикальной оси.



13.Заполнить матрицу случайными числами. Развернуть матрицу на 180º.



14.Заполнить матрицу случайными числами. Отобразить нижнюю половину матрицы на верхнюю зеркально симметрично относительно горизонтальной оси.

15.Заполнить матрицу случайными числами. На побочной диагонали разместить суммы элементов соответствующих столбцов.

5.Список индивидуальных заданий лабораторной работы № 15

1. Дана квадратная матрица. Найти в каждой строке матрицы максимальный элемент и поменять его местами с первым элементом строки.
2. Упорядочить по возрастанию элементы каждой строки матрицы размером $n \times m$.
3. Дана целочисленная квадратная матрица. Найти в каждой строке наименьший элемент и поменять его местами с элементом главной диагонали.
4. Дана действительная матрица размером $n \times m$, все элементы которой различны. В каждой строке выбирается элемент с наименьшим значением, затем среди этих чисел выбирается наибольшее. Указать индексы элемента с найденным значением.
5. Дана действительная квадратная матрица порядка N (N — нечетное), все элементы которой различны. Найти наибольший элемент среди стоящих на главной и побочной диагоналях и поменять его местами с элементом, стоящим на пересечении этих диагоналей.
6. Для заданной квадратной матрицы сформировать одномерный массив из ее диагональных элементов. Найти след матрицы, суммируя элементы одномерного массива.
7. Заданы матрица порядка n и число k. Разделить элементы k-й строки на диагональный элемент, расположенный в этой строке.
8. Для целочисленной квадратной матрицы найти число элементов, кратных k и наибольший из них.
9. Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами.
10. Дана прямоугольная матрица. Найти строку с наибольшей и наименьшей суммой элементов. Вывести на печать найденные строки и суммы их элементов.
11. В данной действительной квадратной матрице порядка n найти сумму элементов строки, в которой расположен элемент с наименьшим значением. Предполагается, что такой элемент единствен.
12. В данной действительной квадратной матрице порядка n найти наибольший по модулю элемент. Получить квадратную матрицу порядка n—1 путем отбрасывания в исходной

матрице строки и столбца, на пересечении которых расположен элемент с найденным значением.

13. Вычислить сумму и число положительных элементов матрицы *A[N,N]* находящихся над главной диагональю.
14. Задана квадратная матрица. Поменять местами строку с максимальным элементом на главной диагонали со строкой с заданным номером *m*.
15. Дана целочисленная прямоугольная матрица. Определить количество строк, не содержащих ни одного нулевого элемента.

6.Пример выполнения лабораторной работы №14

6.1. Условие задачи

Заполнить случайными целыми числами квадратную матрицу. Разместить в последнем столбце произведения элементов соответствующих строк.

6.2.Используемые переменные

`n` –количество строк и столбцов в массиве;

`p` – произведение элементов строки массива;

`x[i][j]` – элементы массива;

i, j – индексы, используемые в циклах.

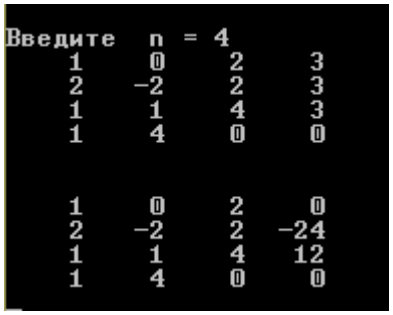
6.3.Текст программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#include<conio.h>// для задержки экрана
#include<ctime>// для работы со случайными числами
#include<iomanip>
using namespace std;  // использовать пространство имён std
int main(){              // заголовок главной функции
   setlocale(LC_ALL,".1251");  //  возможность  переключения на
русский язык
int i,j,n;
   srand(time(0));    // инициализация датчика случайных чисел
float p,x[10][10];
cout<<"\nВведите  n = ";
cin>>n;             //  размерностьмассива
for(i=0; i<n; i++){
   for(j=0; j<n; j++)
{  x[i][j]=rand()%8-3;  //  заполнение массива случайными числами
          cout<<setw(5)<<x[i][j];  //  вывод массива на экран
}
   cout<<endl;
   }
cout<<endl<<endl;
```

```
for(i=0; i<n; i++){
       p=1;
    for(j=0; j<n; j++)
{   p=p*x[i][j];
        }
    x[i][n-1]=p;
    }
    for(i=0; i<n; i++){
    for(j=0; j<n; j++)
cout<<setw(5)<<x[i][j];  //  вывод преобразованного массива
       cout<<endl;
     }
     getch();
}
```

6.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:



7.Пример выполнения лабораторной работы №15

7.1. Условие задачи

Определить номера строк матрицы *R[M,N],* хотя бы один элемент которых равен *c*, и элементы этих строк умножить на *d.*

7.2.Используемые переменные

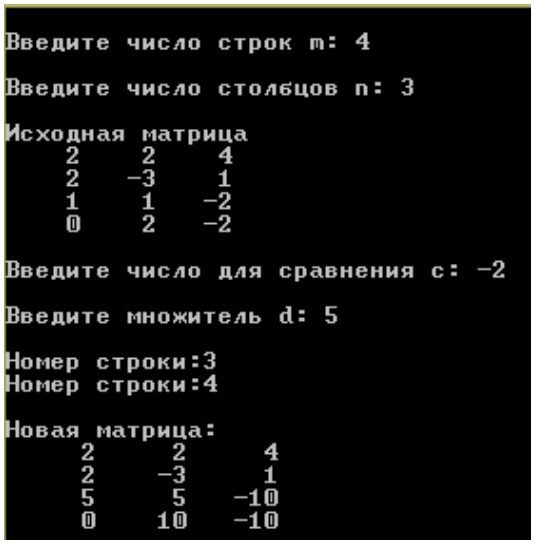
`m` – количество строк в массиве;
n– количество столбцов в массиве;
`A[i][j]` – элементы массива;
`i, j` – индексы, используемые в циклах;
`c` – число для сравнения;
`d` – множитель;
`B[ i ]` – вспомогательный одномерный массив.

7.3.Текст программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#include<conio.h>// для задержки экрана
#include<ctime>// для работы со случайными числами
#include<iomanip>
using namespace std;  // использовать пространство имён std
int main(){            // заголовок главной функции
   setlocale(LC_ALL,".1251");    //  возможность  переключения на
русский язык
   srand(time(0));    // инициализация датчика случайных чисел

int i,j,m,n,A[10][10],c,d,B[10];

cout<<"\nВведите число строк m: ";
   cin>>m;
   cout<<"\nВведите число столбцов n: ";
   cin>>n;
cout<<"\nИсходная матрица "<<endl;
for(i=0;i<m;i++) B[i]=m;
for(i=0; i<m; i++){
   for(j=0; j<n; j++)
{  A[i][j]=rand()%8-3;  //  заполнение массива случайными числами

      }
      cout<<endl;
   }
   cout<<"\nВведите число для сравнения c: ";
   cin>>c;
   cout<<"\nВведите множитель d: ";
cin>>d;
for(i=0;i<m;i++)
{  for(j=0; j<n; j++)
if(A[i][j]==c)
    B[i]=i;
   }
for(i=0;i<m;i++)
if(B[i]==i)
{  cout<<"\nНомер строки:"<<i+1;
for(j=0; j<n; j++)
      A[i][j]*=d;
}
   cout<<"\nНовая матрица:\n";
for(i=0;i<m;i++)
{  for(j=0; j<n; j++)
cout<<setw(6)<<A[i][j];
cout<<endl;
   }
   getch();
}
```

7.4.Тестирование программы

Результат выполнения программы может выглядеть следующим образом:



ЛАБОРАТОРНАЯ РАБОТА №17. РАБОТА СО СТРОКАМИ В ЯЗЫКЕ С++.

1. **Краткие сведения из теории**

В языке С++ нет специально определенного строкового типа данных. Символьные строки организуются как массивы символов, последним из которых является символ \0, внутренний код которого равен нулю. На длину символьного массива в С++ нет ограничения.
Строка описывается как символьный массив. Например:

```
char STR[20] ;
```

Одновременно с описанием строка может инициализироваться. Возможны два способа инициализации строки — с помощью строковой константы и в виде списка символов:

```
char S[10]="строка";
char S []="строка";
char S[10]={'с','т*,'р','о','к','а','\0'};
```

По результату первого описания под строку S будет выделено 10 байт памяти, из них первые 7 получат значения при инициализации (седьмой — нулевой символ). Второе описание сформирует строку из семи символов. Третье описание по результату равнозначно первому. Конечно, можно определить символьный массив и так:

```
char S[10]={'с','т','р','о','к','а'};
```

т. е. без нулевого символа в конце. Но это приведет к проблемам с обработкой такой строки, так как будет отсутствовать ориентир на его окончание.
Отдельные символы строки идентифицируются индексированными именами. Например, в описанной выше строке `S [0] =' с', S[5]='а'.`
Обработка строк обычно связана с перебором всех символов от начала до конца. Признаком конца такого перебора является обнаружение нулевого символа. В следующей программе производятся последовательная замена всех символов строки на звездочки и подсчет длины строки.
Пример 1.

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык

 using namespace std;  // использовать пространство имён std
void main() {           // заголовок главной функции
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    char S[] = "fh5j";
    int i = 0;
    puts(S);
    while (S[i]) {
        S[i++] = '*';
        puts(S);
    }
    printf("\nДлина строки = %d", i);
    printf("\n");
}
```

В результате выполнения программы на экране получим:

```
fh5j
*h5j
**5j
***j
```

```
****
Длина строки=4
```

В этой программе цикл повторяет свое выполнение, пока `S[i]` не получит значение нулевого символа.
Для вывода строки на экран имеется функция `puts()`. Аргументом этой функции указывается имя строки. В этой же библиотеке есть функция ввода строки с клавиатуры с именем `gets()`. В качестве аргумента указывается имя строки, в которую производится ввод.
Среди стандартных библиотек C++ существует библиотека функций для обработки строк. Ее заголовочный файл — `string.h`. В следующем примере используется функция определения длины строки из этой библиотеки. Имя функции — `strlen()`. В качестве аргумента указывается имя строки.
Пример 2. Ввести символьную строку. Перевернуть (обратить) эту строку. Например, если ввели строку «`abcdef`», то в результате в ней должны получить «`fedcba`».

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык

 using namespace std;  // использовать пространство имён std
void main() {           // заголовок главной функции
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    char C, S[10];
    int i;
    printf("Введите строку:  ");
    gets_s(S);
    for (i = 0; i <= (strlen(S) - 1) / 2; i++) {
        C = S[i]; S[i] = S[strlen(S) - i - 1];
        S[strlen(S) - i - 1] = C;
    }
    printf("\nПеревёрнутая строка:  ");
    puts(S);
}
```

Идея алгоритма состоит в перестановке символов, расположенных на одинаковом расстоянии от начала и конца строки. Перебор элементов строки доходит до ее середины. Составляя подобные программы, не надо забывать, что индекс первого символа строки — 0, а индекс последнего на единицу меньше длины строки.

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.

5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Каким образом определяются строки в C++?

2. Какие объекты используются для ввода и вывода строк?

3. Какую библиотеку нужно подключить для использования функций работы со строками?

4. Перечислите простейшие функции работы со строками.

5. Что такое инициализация строки, и как она может быть реализована?

4.Задание к выполнению

Составить блок-схему алгоритма и написать программу, соответствующую номеру вашего варианта:

1. Дана строка, заканчивающаяся точкой. Подсчитать, сколько слов в строке.
2. Дана строка, содержащая английский текст. Найти количество слов, начинающихся с буквы *b.*
3. Дана строка. Подсчитать, сколько в ней букв *k, t*.
4. Дана строка символов, среди которых есть двоеточие (:). Определить, сколько символов ему предшествует.
5. Дана строка. Преобразовать ее, удалив каждый символ * и повторив каждый символ, отличный от *.
6. Дана строка. Подсчитать количество букв *k* в последнем ее слове.
7. Дана строка символов, среди которых есть одна открывающаяся и одна закрывающаяся скобка. Вывести на экран все символы, расположенные внутри этих скобок.
8. В строке заменить все двоеточия (:) точкой с запятой (;). Подсчитать количество замен.
9. В строке между словами вставить вместо пробела запятую и пробел.
10. Удалить часть символьной строки, заключенной в скобки (вместе со скобками).
11. В строке имеется одна точка с запятой (;). Подсчитать количество символов до точки с запятой и после нее.
12. Дана строка. Преобразовать ее, заменив точками все двоеточия (:), встречающиеся среди первых *n/2* символов, и заменив точками все восклицательные знаки, встречающиеся среди символов, стоящих после *n/2* символов.
13. Строка содержит одно слово. Проверить, будет ли оно читаться одинаково справа налево и слева направо (т.е. является ли оно палиндромом).
14. Проверить, одинаковое ли число открывающихся и закрывающихся скобок в данной строке.
15. В строке удалить символ «двоеточие» (:) и подсчитать количество удаленных символов.

5.Пример программы

5.1.Условие задачи.

Дан текст. Напишите программу, определяющую процентное отношение строчных и прописных букв к общему числу символов в нем.

5.2. Описание переменных

a – нижняя граница интеграла;

S[i] – исходная строка;

S1[i] – преобразованная строка;
i – индекс;
j – количество заглавных букв;
k – количество строчных букв;
n – общее количество символов в строке.

5.3.Текст программы

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык
using namespace std;  // использовать пространство имён std
int main() {           // заголовок главной функции
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    char S[100], S1[100];
    int i, j = 0, k = 0, n;
    cout << "Введите строку:  ";
    gets_s(S);
    n = strlen(S);          //Вычисление длины строки
    cout << "\nДлина строки n=" << n;

    strcpy_s(S1, S);         //Копируем строку   S  в строку  S1
    _strlwr_s(S);               //Преобразует буквы верхнего регистра в строке в соответствующие
                                //           буквы нижнего регистра
    for (i = 0; i <= n; i++)
        if (S[i] != S1[i])  j++;     //Подсчитываем число преобразованных (т.е. заглавных букв)
    cout << "\nКоличество заглавных букв:  " << j;
    cout << "      " << j * 100. / n << "%";

    strcpy_s(S, S1);          //Копируем строку   S1  в строку  S
    _strupr_s(S1);              //Преобразует буквы нижнего регистра в строке в соответствующие
                                //           буквы верхнего регистра
    for (i = 0; i <= n; i++)
        if (S[i] != S1[i])  k++;  //Подсчитываем число преобразованных (т.е. строчных букв)
    cout << "\nКоличество строчных букв:  " << k;
    cout << "      " << k * 100. / n << "%";
}
```

5.4.Тестирование программы

Результат работы программы может выглядеть следующим образом:

```
Введите строку:  AAAss

Длина строки n=5
Количество заглавных букв:  3      60%
Количество строчных букв:  2      40%
```

```
Введите строку:  AA345, ss

Длина строки n=9
Количество заглавных букв:  2      22.2222%
Количество строчных букв:  2      22.2222%
```

Во втором случае заглавные и строчные буквы в сумме дают не 100%, так как кроме букв в строке присутствуют другие символы: цифры, пробел и запятая.

ЛАБОРАТОРНАЯ РАБОТА №17. ФУНКЦИИ-ПОДПРОГРАММЫ В ЯЗЫКЕ С++.

1.Краткие сведения из теории

Определение функции. Обращение к функции. В С++ используется лишь один тип подпрограмм — функция. Здесь вообще не принято употреблять термин «подпрограмма», потому что *функция является основной программной единицей в С++,* минимальным исполняемым программным модулем. Всякая программа обязательно включает в себя основную функцию с именем **main**. Если в программе используются и другие функции, то они выполняют роль подпрограмм.

Рассмотрим пример. Требуется 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).*

Вот программа решения этой задачи с использованием вспомогательной функции.

Пример 1.

```
#include <iostream.h>
//Определение вспомогательной функции
int MAX(int x, int y)
{ if (x>y) return x;
  else return y;
}
//Основная функция
void 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;
}
```

Формат определения функции следующий:

```
тип имя_функции (спецификация_параметров)
{
тело_ функции
}
```

Тип функции — это тип возвращаемого функцией результата. Если функция не возвращает никакого результата, то для нее указывается тип void.

Имя функции — идентификатор, задаваемый программистом или main для основной функции.

Спецификации параметров — это либо «пусто», либо список имен формальных параметров функции с указанием типа для каждого из них.

Тело функции — это либо составной оператор, либо *блок.*

Здесь мы впервые встречаемся с понятием блока. Признаком блока является наличие описаний программных объектов (переменных, массивов и т.д.), которые действуют в пределах этого блока. Блок, как и составной оператор, ограничивается фигурными скобками.

В С++ действует правило: *тело функции не может содержать в себе определения других функций.* Иначе говоря, недопустимы внутренние функции, как это делается в Паскале. Из всякой функции возможно обращение к другим функциям, однако они всегда являются внешними по отношению к вызывающей.

Оператором возврата из функции в точку ее вызова является оператор return. Он может использоваться в функциях в двух формах:

return; или **return** выражение;

В первом случае функция не возвращает никакого значения в качестве своего результата. Во втором случае результатом функции является значение указанного выражения. Тип этого выражения должен либо совпадать с типом функции, либо относиться к числу типов, допускающих автоматическое преобразование к типу функции.

Оператор return может в явном виде отсутствовать в теле функции. В таком случае его присутствие подразумевается перед закрывающей тело функции фигурной скобкой. Такая подстановка производится компилятором.

Формат обращения к функции (вызова функции) традиционный:

имя_функции(список_фактических_параметров)

Однако в С++ обращение к функции имеет своеобразную трактовку: обращение к функции — это выражение. В этом выражении круглые скобки играют роль знака операции, для которой функция и фактические параметры (аргументы) являются операндами. Приоритет операции «скобки» самый высокий, поэтому вычисление функции в выражениях производится раньше других операций.

Между формальными и фактическими параметрами при вызове функции должны соблюдаться правила соответствия *по последовательности* и *по типам.* Фактический параметр — это выражение того же типа, что и у соответствующего ему формального параметра. Стандарт языка Си допускает автоматическое преобразование значений фактических параметров к типу формальных параметров. В Си++ такое преобразование не предусмотрено. Поэтому в дальнейшем мы будем строго следовать принципу соответствия типов.

Необходимо усвоить еще один важнейший принцип, действующий в С++: *передача параметров при вызове функции происходит только по значению.* Поэтому *выполнение функции не может изменить значения переменных, указанных в качестве фактических параметров.*

Прототип функции. Оказывается, совсем не обязательно было в предыдущем примере помещать полное определение функции MAX() перед основной частью программы. Вот другой вариант программы, решающей ту же самую задачу.

Пример 2.

```
#include <iostream.h>
//Прототип функции MAX
int MAX(int, int);
//Основная функция
void 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;
}
//Определение функции MAX
int MAX(int x, int y)
{ if (x>y) return x;
  else return y;
}
```

Здесь использован *прототип* функции. Прототипом называется предварительное описание функции, в котором содержатся все необходимые сведения для правильного

обращения к ней: имя и тип функции, типы формальных параметров. В прототипе имена формальных параметров указывать необязательно (как это сделано в примере 2), хотя их указание не является ошибочным. Можно было написать и так, как в заголовке определения функции:

```
int MAX(int x, int y);
```

Точка с запятой в конце прототипа ставится обязательно! Можно было бы записать прототип и в теле основной функции наряду с описаниями других программных объектов в ней.

В следующей программе приводится пример использования функции, которая не имеет параметров и не возвращает никаких значений в точку вызова. Эта функция рисует на экране строку, состоящую из 80 звездочек.

Пример 3.

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

Может возникнуть вопрос: если основная часть программы является функцией, то кто (или что) ее вызывает? Ответ состоит в следующем: программу вызывает операционная система при запуске программы на исполнение. И в принципе `main`-функция совсем не обязательно должна иметь тип `void`. Например, она может возвращать операционной системе целое значение 1 в качестве признака благополучного завершения программы и 0 — в «аварийном» случае. Обработка этих сообщений будет осуществляться системными средствами.

Массив как параметр функции.

Пример 1. Составим программу решения следующей задачи. Дана вещественная матрица `A[M][N]`. Требуется вычислить и вывести евклидовы нормы строк этой матрицы.

Евклидовой нормой вектора называют корень квадратный из суммы квадратов его элементов:

$$L = \sqrt{\sum_{i=1}^{n} x_i^2}\ .$$

Если строку матрицы рассматривать как вектор, то данную формулу надо применить к каждой строке. В результате получим *M* чисел.

Определение функции вычисления нормы произвольного вектора:

```
double Norma(int n, double X[])
{ int i;
   double S=0;
   for(i=0; i<n; i++) S+=X[i]*X[i];
   return sqrt(S);
}
```

Заголовок этой функции можно было записать и в такой форме:

```
double Norma(int n, double *X)
```

В обоих случаях в качестве второго параметра функции используется указатель на начало массива. Во втором варианте это более очевидно, однако оба варианта тождественны.

При вызове функции `Norma()` в качестве второго фактического параметра должен передаваться адрес начала массива (вектора).

Рассмотрим фрагмент основной программы, использующей данную функцию для обработки матрицы размером 5 x 10.

```
void main()
{  double A[5][10];  int i;
   //Ввод матрицы
. . . . . .
   //Вычисление и вывод нормы строк
   for(i=0; i<5; i++)
   cout<<"Норма"<<i<<"-й строки=" <<Norma(10,A[i]);
}
```

В обращении к функции второй фактический параметр `A[i]` является указателем на начало *i*-й строки матрицы `A`.

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

4.Вопросы к защите

1. Каков формат определения функций в C++?
2. Что такое прототип функции?
3. Для чего нужны функции-подпрограммы?
4. Может ли массив быть параметром функции?
5. Как отражаются функции в блок-схемах?

5.Задание к выполнению

Составить блок-схему алгоритма и написать программу с использованием функции – подпрограммы обработки массива заданной размерности. Использовать эту функцию в основной программе, обращаясь к ней столько раз, сколько массивов дано в индивидуальном задании.

1. Вычислить суммы положительных элементов для каждого из массивов X(N), Y(M), Z(K).

2. Найти наибольшие элементы и их порядковые номера массивов X(N) и Y(M).

3. Вычислить среднее арифметическое положительных элементов для каждого из массивов A(N1), B(N2), C(N3).

4. Вычислить Z = (S1 + S2) / 2, где S1 – сумма отрицательных элементов массива X(N1), S2 - сумма отрицательных элементов массива Y(N2).

5. Вычислить суммы отрицательных элементов для каждого из массивов X(N), Y(M), Z(K).

6. Вычислить среднее арифметическое отрицательных элементов для каждого из массивов A(N1), B(N2), C(N3).

7. Вычислить Z = (X1 + X2) / 2, где X1 и X2 – наибольшие элементы массивов X1(N), X2(M).

8. Вычислить количество положительных элементов для каждого из массивов X(N), Y(M), Z(K).

9. Найти наименьшие элементы и их порядковые номера массивов X(N) и Y(M).

10. Вычислить среднее арифметическое четных элементов для каждого из массивов A(N1), B(N2), C(N3).

11. Вычислить Z = (S1 + S2) / 2, где S1 – сумма нечетных элементов массива X(N1), S2 - сумма нечетных элементов массива Y(N2).

12. Вычислить количество отрицательных элементов для каждого из массивов X(N), Y(M), Z(K).

13. Вычислить среднее арифметическое элементов для каждого из массивов A(N1), B(N2), C(N3).

14. Вычислить Z = (X1 + X2) / 2, где X1 и X2 – наименьшие элементы массивов X1(N), X2(M).
15. Вычислить произведение отрицательных элементов для каждого из массивов X(N), Y(M), Z(K).

6.Пример выполнения задания

6.1.Условие задачи.

Даны два массива: A(N1) и B(N2). Для каждого массива найти элемент, наиболее близкий к среднему арифметическому компонентов массива.

6.2. Описание переменных

Главная функция:

A,B – массивы;
N – размерность массива A;
K – размерность массива B;
M1, M2 – значения элементов, наиболее близких к среднему арифметическому массива.

Функция заполнения массива и вывода на экран **`Matr(int n,int M[])`**

M – массив;
n – размерность массива;
i – индекс.

Функция нахождения элемента, наиболее близкого к среднему арифметическому
`min1(int L,int V[])`

V– массив;
L– размерность массива;
i – индекс;
sr – среднее арифметическое;
min – разность между средним арифметическим и текущим элементом массива.

6.3.Текст программы

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык
#include <ctime>    // для работы со случайными числами
using namespace std;  // использовать пространство имён std

// Функция заполнения массива и вывода на экран
void Matr(int n, int M[])
{
    int i;
    srand(time(0));    // инициализация датчика случайных чисел
    for (i = 0; i < n; i++)
    {
        M[i] = rand() % 10;
        cout << "  "<< M[i];
    }
}
```

```
float min1(int L, int V[]) // Вычисление элемента, наиболее близкого
                           // к среднему арифметическому
{
    float sr=0,min,m=V[0];
    for (int i = 0; i < L; i++)
        sr += V[i];
    sr /= L;
    cout << "\n sr = " << sr;
    min = abs(sr - V[0]);
    for (int i = 0; i < L; i++)
        if (abs(sr - V[i]) < min)
        {
            min = abs(sr - V[i]); m = V[i];
        }
    return m;
}
int main()     //Главная функция
{
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    int A[20], B[20];
    int N, K;
    float M1, M2;
    cout << " Введите размерность первого массива: "; cin >> N;
    Matr(N, A);
    M1 = min1(N, A);
    cout << "\n   M1=" << M1 << endl;
    cout << " Введите размерность второго массива:  "; cin >> K;
    Matr(K, B);
    M2 = min1(K, B);
    cout << "\n   M2=" << M2 << endl;

}
```

6.4.Тестирование программы

Результат работы программы может выглядеть следующим образом:

```
Введите размерность первого массива: 3
 3  3  4
sr = 3.33333
  M1=3
Введите размерность второго массива:  4
 9  7  1  0
sr = 4.25
  M2=7
```

ЛАБОРАТОРНАЯ РАБОТА №18. ДВУМЕРНЫЙ МАССИВ КАК ПАРАМЕТР ФУНКЦИИ

1.Краткие сведения из теории

Массив как параметр функции.

Пример 1. Составим программу решения следующей задачи. Дана вещественная матрица `A[M][N]`. Требуется вычислить и вывести евклидовы нормы строк этой матрицы.

Евклидовой нормой вектора называют корень квадратный из суммы квадратов его элементов:

$$L = \sqrt{\sum_{i=1}^{n} x_i^2}\ .$$

Если строку матрицы рассматривать как вектор, то данную формулу надо применить к каждой строке. В результате получим *M* чисел.

Определение функции вычисления нормы произвольного вектора:

```
double Norma(int n, double X[])
{ int i;
   double S=0;
   for(i=0; i<n; i++) S+=X[i]*X[i];
   return sqrt(S);
}
```

Заголовок этой функции можно было записать и в такой форме:

```
double Norma(int n, double *X)
```

В обоих случаях в качестве второго параметра функции используется указатель на начало массива. Во втором варианте это более очевидно, однако оба варианта тождественны.

При вызове функции `Norma()` в качестве второго фактического параметра должен передаваться адрес начала массива (вектора).

Рассмотрим фрагмент основной программы, использующей данную функцию для обработки матрицы размером 5 x 10.

```
void main()
{  double A[5][10];  int i;
   //Ввод матрицы
. . . . . .
   //Вычисление и вывод нормы строк
   for(i=0; i<5; i++)
   cout<<"Норма"<<i<<"-й строки=" <<Norma(10,A[i]);
}
```

В обращении к функции второй фактический параметр `A[i]` является указателем на начало *i*-й строки матрицы `A`.

Пример 2. Заполнить двумерную матрицу случайными целыми числами в диапазоне от 0 до 99. Отсортировать строки полученной матрицы по возрастанию значений. Отсортированную матрицу вывести на экран.

```
#include <iostream.h>
#include <iomanip.h>
```

```
#include <conio.h>
#include <stdlib.h>
const n=5;//Глобальное объявление константы
//Прототипы функций
void Matr(int M[][n]);
void Sort(int, int X[]);
//Основная программа
void main()
{
  int i,j, A[n][n];
  clrscr();
  cout<<"\n"<<"Матрица до сортировки:"<<"\n";
  Matr(A);
  for(i=0; i<n; i++) Sort(n, A[i]);
  cout<<"\n"<<"Матрица после сортировки:"<<"\n";
  for(i=0; i<n; i++)
  {  for(j=0; j<n; j++)
     cout<<setw(6)<<A[i][j];
     cout<<endl;}
}
// Функция сортировки вектора
void Sort(int k, int X[])
{int i,j, Y;
 for(i=0; i<k-1; i++)
   for(j=0; j<k-i-1; j++)
    if(X[j]>X[j+1]) {Y=X[j]; X[j]=X[j+1];
X[j+1]=Y;}
}
 //Функция заполнения матрицы и вывода на экран
 void Matr(int M[][n])
 {int i,j;
  randomize(); //Установка датчика случайных чисел
  for(i=0; i<n; i++)
  {  for(j=0; j<n; j++)
     { M[i][j]=rand()%100; cout<<setw(6)<<M[i][j];}
     cout<<endl;
   }
 }
```

Обратите внимание на прототип и заголовок функции `Matr()` . В них явно указывается вторая размерность параметра-матрицы. Первую тоже можно указать, но это необязательно. Как уже говорилось выше, двумерный массив рассматривается как одномерный массив, элементами которого являются массивы (в данном случае — строки матрицы). Компилятору необходимо «знать» размер этих элементов. Для массивов большей размерности (3, 4 и т.д.) в заголовках функций необходимо указывать все размеры, начиная со второго.

При обращении к функции `Matr()` фактическим параметром является указатель на начало двумерного массива `A`, а при обращении к функции `Sort ()` — указатели на начало строк.

В итоге тестирования программы получен следующий результат.

Матрица до сортировки:

```
46    23    57    35    18
 8    48    68     4    70
56    98    16    71    40
70    84    66    67    11
20    44    37    57    38
```

Матрица после сортировки:

18	23	35	46	57
4	8	48	68	70
16	40	56	71	98
11	66	67	70	84
20	37	38	44	57

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

9.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Каков формат определения функций в C++?
2. Что такое прототип функции?
3. Для чего нужны функции-подпрограммы?
4. Может ли двумерный массив быть параметром функции?
5. Как отражаются функции в блок-схемах?

4.Задание к выполнению

Составить блок-схему алгоритма и написать программу с использованием функции – подпрограммы обработки массива заданной размерности. Использовать эту функцию в основной программе, обращаясь к ней столько раз, сколько массивов дано в индивидуальном задании.

1. Подсчитать число нулевых элементов для каждой из матриц A(N,M) и B(M,N).
2. Вычислить суммы элементов главной диагонали каждой из матриц A(N,N) и B(M,M).
3. Найти наибольшие элементы и их номера для каждой из матриц A(N1,N2) и B(M1,M2).
4. Вычислить среднее арифметическое модулей отрицательных элементов для каждой из матриц A(N,M) и B(M,N).
5. Подсчитать количество элементов матриц X(N1,N2) и Y(M1,M2), удовлетворяющих условиям: $0 <= X(i, j) <=1$, $0 <= Y(i, j) <=1$.
6. Подсчитать число отрицательных элементов для каждой из матриц A(N,M) и B(M,N).

7. Подсчитать количество элементов матриц X(M1,M2) и Y(N1,N2), удовлетворяющих условиям: 1 <= X(i, j) <=2, 1 <= Y(i, j) <=2.
8. Вычислить Z = (X1 + X2) / 2, где X1 и X2 – наибольшие элементы массивов X1(N1,N2), X2(M1,M2).
9. Подсчитать сумму отрицательных элементов для каждой из матриц A(N,M) и B(M,N).
10. Вычислить суммы элементов побочной диагонали каждой из матриц A(N,N) и B(M,M).
11. Найти наименьшие элементы и их номера для каждой из матриц A(N1,N2) и B(M1,M2).
12. Вычислить среднее арифметическое модулей четных элементов для каждой из матриц A(N,M) и B(M,N).
13. Подсчитать количество нечетных элементов матриц X(N1,N2) и Y(M1,M2).
14. Подсчитать сумму отрицательных элементов для каждой из матриц A(N,M) и B(M,N).
15. Вычислить Z = (X1 + X2) / 2, где X1 и X2 – наименьшие элементы массивов X1(N1,N2), X2(M1,M2).

5.Пример программы

5.1.Условие задачи.

Подсчитать для каждой из матриц A(N,M) и B(M,N) количество элементов, кратных трём.

5.2.Текст программы

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык
#include <ctime>    // для работы со случайными числами
#include<iomanip>  // для подключения манипуляторов
using namespace std;  // использовать пространство имён std

// Прототипы функций
void Matr(int, int, int M[][20]);
int Kol(int, int, int M[][20]);

// Основная программа
int main(){
    int i, j,n,m, A[20][20];
    setlocale(LC_ALL, ".1251");    //  возможность  переключения на русский язык
    for (int i = 1; i <= 2; i++) {
        cout << "\n Введите количество строк m = ";  cin >> m;
        cout << " Введите количество столбцов n = ";  cin >> n;

        cout << "\n Матрица: \n";
        Matr(m, n, A);
        int K = Kol(m, n, A);
        if (K == 0) cout << "\n  Нет элементов кратных 3 \n";
        else cout << "\n Количество элементов, кратных трем, K = " << K << "\n";
    }
}
```

```
// Функция заполнения матрицы и вывода на экран
void Matr(int M, int N, int X[][20]) {
    int i, j;
    srand(time(0)); // инициализация датчика случайных чисел
    for (i = 0; i < M; i++) {
        for (j = 0; j < N; j++)
        {
            X[i][j] = rand() % 31;  //  заполнение массива случайными числами
            cout << setw(6) << X[i][j];  //  вывод массива на экран
        }
        cout << endl;
    }
}
    // Функция вычисления количества элементов, кратных 3
int Kol(int M, int N, int X[][20]) {
        int K = 0;
        for (int i = 0; i < M; i++)
            for (int j = 0; j < N; j++)
                if (X[i][j] % 3 == 0) K++;
        return K;
}
```

5.3.Тестирование программы

Результат работы программы может выглядеть следующим образом:

```
Введите количество строк m = 3
Введите количество столбцов n = 2

Матрица:
     6     4
     2    21
    11    30

Количество элементов, кратных трем, K = 3

Введите количество строк m = 1
Введите количество столбцов n = 2

Матрица:
    16     8

 Нет элементов кратных 3
```

ЛАБОРАТОРНАЯ РАБОТА №19. РАБОТА С ФАЙЛАМИ НА ЯЗЫКЕ С++

1.Краткие сведения из теории

Для работы с файлами в стандартной библиотеке определен заголовочный файл **fstream**, который определяет базовые типы для чтения и записи файлов. В частности, это:

- **ifstream**: для чтения с файла
- **ofstream**: для записи в файл
- **fstream**: совмещает запись и чтение

Для работы с данными типа **wchar_t** для этих потоков определены двойники:

- **wifstream**
- **wofstream**
- **wfstream**

Открытие файла

При операциях с файлом вначале необходимо открыть файл с помощью функции **open()**. Данная функция имеет две версии:

- **open(путь)**
- **open(путь, режим)**

Для открытия файла в функцию необходимо передать путь к файлу в виде строки. И также можно указать режим открытия. Список доступных режимов открытия файла:

- **ios::in**: файл открывается для ввода (чтения). Может быть установлен только для объекта ifstream или fstream
- **ios::out**: файл открывается для вывода (записи). При этом старые данные удаляются. Может быть установлен только для объекта ofstream или fstream
- **ios::app**: файл открывается для дозаписи. Старые данные не удаляются.
- **ios::ate**: после открытия файла перемещает указатель в конец файла
- **ios::trunc**: файл усекается при открытии. Может быть установлен, если также установлен режим out
- **ios::binary**: файл открывается в бинарном режиме

Если при открытии режим не указан, то по умолчанию для объектов ofstream применяется режим ios::out, а для объектов ifstream – режим ios::in. Для объектов fstream совмещаются режимы ios::out и ios::in.

```
ofstream out;          // поток для записи
out.open("D:\\hello1.txt"); // открываем файл для записи

ofstream out2;
out2.open("D:\\hello2.txt", ios::app); // открываем файл для дозаписи

ofstream out3;
out2.open("D:\\hello3.txt", ios::out | ios::trunc); // установка нескольких режимов

ifstream in;       // поток для чтения
in.open("D:\\hello4.txt"); // открываем файл для чтения
```

```
fstream fs;         // поток для чтения-записи
fs.open("D:\\hello5.txt"); // открываем файл для чтения-записи
```

Однако в принципе необязательно использовать функцию `open` для открытия файла. В качестве альтернативы можно также использовать конструктор объектов-потоков и передавать в них путь к файлу и режим открытия:

`fstream(путь)`

`fstream(путь, режим)`

При вызове конструктора, в который передан путь к файлу, данный файл будет автоматически открываться:

```
ofstream out("D:\\hello.txt");
ifstream in("D:\\hello.txt");
fstream fs("D:\\hello.txt", ios::app);
```

Вообще использование конструкторов для открытия потока является более предпочтительным, так как определение переменной, представляющей файловой поток, уже предполагает, что этот поток будет открыт для чтения или записи. А использование конструктора избавит от ситуации, когда мы забудем открыть поток, но при этом начнем его использовать.

В процессе работы мы можем проверить, открыт ли файл с помощью функции `is_open()`. Если файл открыт, то она возвращает `true`:

```
ifstream in;        // поток для чтения
in.open("D:\\hello.txt"); // открываем файл для чтения
// если файл открыт
if (in.is_open())
{
}
```

Закрытие файла

После завершения работы с файлом его следует закрыть с помощью функции **`close()`**. Также стоит отметить, то при выходе объекта потока из области видимости, он удаляется, и у него автоматически вызывается функция `close`.

```
#include <iostream>
#include <fstream>
int main()
{
    std::ofstream out;          // поток для записи
    out.open("D:\\hello.txt"); // открываем файл для записи
    out.close();               // закрываем файл
    std::ifstream in;          // поток для чтения
    in.open("D:\\hello.txt"); // открываем файл для чтения
    in.close();                // закрываем файл
    std::fstream fs;           // поток для чтения-записи
    fs.open("D:\\hello.txt"); // открываем файл для чтения-записи
    fs.close();                // закрываем файл

    return 0;
}
```

Запись в файл

Для записи в файл к объекту `ofstream или fstream` применяется оператор **<<** (как и при выводе на консоль):

```
#include <iostream>
#include <fstream>
int main()
{   std::ofstream out;          // поток для записи
    out.open("D:\\hello.txt"); // окрываем файл для записи
     if (out.is_open())
     {
         out << "Hello World!" << std::endl;
     }
     std::cout << "End of program" << std::endl;
     return 0;
}
```

Данный способ перезаписывает файл заново. Если надо дозаписать текст в конец файла, то для открытия файла нужно использовать режим **`ios::app`**:

```
std::ofstream out("D:\\hello.txt", std::ios::app);
if (out.is_open())
{
    out << "Welcome to CPP" << std::endl;
}
out.close();
```

Чтение из файла

Если надо считать всю строку целиком или даже все строки из файла, то лучше использовать встроенную функцию **`getline()`**, которая принимает поток для чтения и переменную, в которую надо считать текст:

```
#include <iostream>
#include <fstream>
#include <cstring>
int main()
{    std::string line;
    std::ifstream in("D:\\hello.txt"); // открываем файл для чтения
    if (in.is_open())
    {
        while (getline(in, line))
        {
            std::cout << line << std::endl;
        }
    }
    in.close();     // закрываем файл
    std::cout << "End of program" << std::endl;
    return 0;
}
```

Также для чтения данных из файла для объектов `ifstream` и `fstream` может применяться оператор >> (также как и при чтении с консоли):

```
#include <iostream>
#include <fstream>
#include <vector>
struct Operation
{   int sum;          // купленная сумма
    double rate;      // по какому курсу
    Operation(double s, double r) : sum(s), rate(r)
    {}
};
int main()
{   std::vector<Operation> operations = {
        Operation(120, 57.7),
        Operation(1030, 57.4),
        Operation(980, 58.5),
        Operation(560, 57.2)
    };
    std::ofstream out("D:\\operations.txt");
    if (out.is_open())
    {
      for (int i = 0; i < operations.size(); i++)
        {
        out<<operations[i].sum<<" "<< operations[i].rate << std::endl;
        }
     }
     out.close();
     std::vector<Operation> new_operations;
     double rate;
     int sum;
     std::ifstream  in("D:\\operations.txt");  //  открываем  файл  для
чтения
     if (in.is_open())
     {
         while (in >> sum >> rate)
         {
             new_operations.push_back(Operation(sum, rate));
         }
     }
     in.close();
     for (int i = 0; i < new_operations.size(); i++)
     {
      std::cout<<new_operations[i].sum<<" - "<< new_operations[i].rate
<< std::endl;
     }
     return 0;
}
Здесь вектор структур Operation записывается в файл.
for (int i = 0; i < operations.size(); i++)
{
    out << operations[i].sum << " " << operations[i].rate << std::endl;
}
```

При записи в данном случае будет создаваться файл в формате

120 57.7

1030 57.4

980 58.5

560 57.2

Используя оператор >>, можно считать последовательно данные в переменные `sum` и `rate` и ими инициализировать структуру.

```
while (in >> sum >> rate)
{
    new_operations.push_back(Operation(sum, rate));
}
```

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

2.2 Содержание отчёта

1. Номер работы и её название.
2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Каков формат объявления файла в C++?
2. Перечислите режимы открытия файлов.
3. Как реализован форматный ввод данных в файл и вывод из файла?

4.Задание к выполнению

Составить блок-схему алгоритма и написать программу в соответствии с номером вашего варианта.

1. Записать в файл N натуральных чисел. Получить в другом файле все компоненты исходного файла кроме тех, которые кратны K. Вывести полученный файл на печать.

2. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Найти среди компонентов файла количество чисел, которые делятся на 2, но не делятся на 4.

3. Записать в файл N действительных чисел. Найти наибольшее из значений модулей компонентов с нечетными номерами.

4. Записать в файл N действительных чисел. Вычислить произведение компонентов файла и вывести на печать.

5. Записать в файл N действительных чисел. Найти разность первого и последнего компонентов файла.

6. Записать в файл f N целых чисел, полученных с помощью генератора случайных чисел. Заполнить файл g числами, которые являются произведениями соседних компонентов файла f .

7. Заполнить файл f натуральными числами, полученными с помощью генератора случайных чисел. Найти количество нечетных чисел среди компонентов.

8. Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g все компоненты файла f, которые делятся на m и не делятся на n.

9. Записать в файл f N целых чисел, полученных с помощью генератора случайных чисел. Подсчитать количество отрицательных чисел среди компонентов этого файла.
10. Записать в файл N произвольных натуральных чисел. Переписать в другой файл те элементы, которые кратны К. Вывести полученный файл на печать.

11. Заполнить файл N действительными числами, полученными с помощью генератора случайных чисел. Найти сумму минимального и максимального элементов этого файла.

12. Записать в файл N целых чисел. Вычислить сумму четных компонентов файла и вывести на печать.

13. Записать в файл N целых чисел. Получить в другом файле все отрицательные компоненты исходного файла. Вывести полученный файл на печать.

14. Записать в файл N действительных чисел. Найти наименьшее из значений компонентов с четными номерами.

15. Записать в файл N действительных чисел. Вычислить сумму модулей компонентов файла и вывести на печать.

5.Пример 1

5.1Условие задачи.

Заполнить файл f целыми числами, полученными с помощью генератора случайных чисел. Получить в файле g те компоненты файла f , которые являются четными.

5.2 Описание переменных

fp – исходный файл;
gp – файл, содержащий чётные числа;
n – количество чисел в исходном файле;
c – числа;
i – индекс.

5.3.Текст программы

Способ 1:

```cpp
#include<iostream>  //  библиотека ввода-вывода
#include<fstream>
#include<locale.h>  // для переключения на русский язык
#include<ctime>
using namespace std;  // использовать пространство имён std
void main()     //Главная функция
{
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    char x, k;
    int i, n, c;
    srand(time(0));
    ofstream fp("vse.txt");
    cout << "Введите количество чисел: "; cin >> n;
    cout << "\n  Файл f:\n";
    for (i = 0; i < n; i++) {
        c = rand() % 200;
            fp << c<<" ";
            cout << c<<" ";
    }
    fp.close();
    cout << endl;

    ifstream fp1("vse.txt");
    ofstream gp("chet.txt");
    cout << "\n  Файл g:\n";

    while (fp1 >> c)
        if(c%2==0){
            gp << c<<" ";
            cout << c << " ";
    }
    gp.close(); fp1.close();
}
```

Способ 2:

```cpp
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык
#include <ctime>    // для работы со случайными числами
using namespace std;  // использовать пространство имён std
void main()     //Главная функция
{
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    FILE* fp, * gp;     //Объявление файлов
    int i, c, n;
    srand(time(0));    // инициализация датчика случайных чисел
    cout << "Введите количество чисел: "; cin >> n;
    cout << "\n Файл f:\n";
    fopen_s(&fp,"vse.txt", "w+");  //Открытие файла fp для записи
    for (i = 0; i < n; i++)
    {
        c = rand() % 200;          //Заполнение файла fp случайными
        fprintf(fp, "%i\n", c);  //числами и вывод этих
        printf("%i\n", c);       //чисел на экран

    }
    printf("\n");
    fclose(fp);                 //Закрытие файла fp
```

```
    fopen_s(&fp,"vse.txt", "r");    //Открытие файла fp для чтения
    fopen_s(&gp,"chet.txt", "w+"); //Открытие файла gp для записи
    cout << "\n Файл g:\n";
    while ((fscanf_s(fp, "%i", &c)) != EOF)
        if ((c % 2) != 1)
        {
            fprintf(gp, "%i\n", c);//Запись в файл gp и вывод на
            printf("%i \n", c);    //экран четных элементов файла fp
        }
    fclose(fp);             //Закрытие файла fp
    fclose(gp);             //Закрытие файла gp

}
```

5.4.Тестирование программы

Результат работы программы может выглядеть следующим образом:

```
Введите количество чисел: 5

 Файл f:
197
22
178
83
182


 Файл g:
22
178
182
```

6.Пример 2

6.1.Условие задачи.

Создать файл, содержащий текст, записанный строчными буквами. Вывести на экран из файла тот же текст, записанный заглавными буквами.

6.2. Описание переменных

fp – исходный файл;
x – буквы в файле;
k – вспомогательная переменная (разность между кодами заглавных и строчных букв);
i – индекс.

6.3.Текст программы

Способ 1:

```
#include<iostream>  //  библиотека ввода-вывода
#include<fstream>
#include<locale.h>  // для переключения на русский язык

using namespace std;  // использовать пространство имён std
void main()     //Главная функция
{
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    char x,k;

    ofstream fp("bukva.txt");
    cout<<"Введите символы. Признак конца - *";
    while ((x = getchar()) != '*')
        fp << x<<" ";
    fp.close();
    cout << endl;

    ifstream in("bukva.txt");
    k = 'A' - 'a';
    while (in >> x) {
        x += k;
        cout << x << " ";
    }
    in.close();
}
```

Способ 2:

```
#include<iostream>  //  библиотека ввода-вывода
#include<locale.h>  // для переключения на русский язык
#include<stdio.h>
using namespace std;  // использовать пространство имён std
void main()     //Главная функция
{
    setlocale(LC_ALL, ".1251"); // возможность переключения на русский язык
    FILE* fp;
    char x, k;
    int i;
    fopen_s(&fp,"buk.txt", "w+");
    puts("Введите символы. Признак конца - *");
    while ((x = getchar()) != '*')
        fprintf(fp, "%c",x);
    printf("\n");
    fclose(fp);
    fopen_s(&fp,"buk.txt", "r");
    k = 'A' - 'a';
    while ((fscanf_s(fp, "%c", &x)) != EOF)
        if (x != '\n')  printf("%c", x + k);
        else printf("\n");
    fclose(fp);
}
```

6.4.Тестирование программы

Результат работы программы может выглядеть следующим образом:

```
Введите символы. Признак конца - *
asdf ghj kl*

ASDF GHJ KL_
```

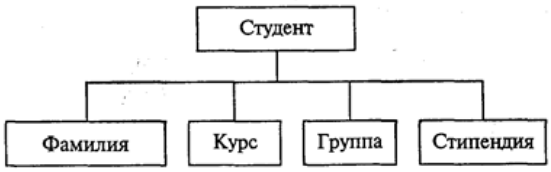
ЛАБОРАТОРНАЯ РАБОТА №20. РАБОТА СО СТРУКТУРАМИ В ЯЗЫКЕ С++

1. Краткие сведения из теории

В языке С++ понятие структуры аналогично понятию записи (`record`) в Паскале. Это структурированный тип данных, представляющий собой поименованную совокупность разнотипных элементов. Тип `структура` обычно используется при разработке информационных систем, баз данных.

Правила использования структур рассмотрим на примере.

Сведения о выплате студентам стипендии требуется организовать в виде, показанном на рис.



Элементы такой структуры (фамилия, курс, группа, стипендия) называются полями. Каждому полю должно быть поставлено в соответствие имя и тип.

Формат описания структурного типа следующий:

```
struct имя_типа
        {определения_элементов};
```

В конце обязательно ставится точка с запятой (это оператор). Для рассмотренного примера определение соответствующего структурного типа может быть следующим:

```
struct student{  char fam[30];
                 int kurs;
                 char grup[3];
                 float stip;
              };
```

После этого, `student` становится именем структурного типа, который может быть назначен некоторым переменным. В соответствие со стандартом С++ это нужно делать так:

```
struct student stud1, stud2;
```

Правила С++ разрешают в этом случае служебное слово `struct` опускать и писать

```
student stud1, stud2;
```

Здесь `stud1` и `stud2` — переменные структурного типа. Допускаются и другие варианты описания структурных переменных. Можно вообще не задавать имя типа, а описывать сразу переменные:

```
struct {char fam[30];
   int kurs;
   char grup[3];
   float stip;
   }
   studi, stud2, *pst;
```

В этом примере кроме двух переменных структурного типа объявлен указатель `pst` на такую структуру. В данном описании можно было сохранить имя структурного типа `student`.

Обращение к элементам (полям) структурной величины производится с помощью

уточненного имени следующего формата:

```
имя_структуры.имя_элемента
```

Примеры уточненных имен для описанных выше переменных:

```
studi.fam;   stud1.stip
```

Значения элементов структуры могут определяться вводом, присваиванием, инициализацией. Пример инициализации в описании:

```
student studl={"Кротов", 3, "Ф32", 350};
```

Пусть в программе определен указатель на структуру

```
student *pst, stud1;
```

Тогда после выполнения оператора присваивания

```
pst=&studl;
```

к каждому элементу структурной переменной studi можно обращаться тремя способами. Например, для поля `fam`

`stud1.fam` или `(*pst).fam` или `pst→fam`

В последнем варианте используется *знак операции доступа к элементу структуры: —>*. Аналогично можно обращаться и к другим элементам этой переменной:

```
pst->FIO, pst->grup, pst->stip.
```

Поля структуры могут сами иметь структурный тип. Такие величины представляют многоуровневые деревья.

Допускается использование массивов структур. Например, сведения о 100 студентах могут храниться в массиве, описанном следующим образом:

```
student stud[100];
```

Тогда сведения об отдельных студентах будут обозначаться, например, так:`stud[l].fam, stud [5].kurs` и т.п. Если нужно взять первую букву фамилии 25-го студента, то следует писать:

```
stud[25].fam[0].
```

2.Порядок выполнения работы

2.1 Последовательность выполнения лабораторных работ

1. Получение у преподавателя варианта задания для предварительной проработки материала дома.
2. Предварительная проработка материала (подготовка необходимых формул и изучение рекомендованного теоретического материала).
3. Написание программы и ее отладка на ПК.
4. Тестирование программы.
5. Составление отчёта по итогам лабораторной работы.
6. Защита лабораторной работы.

10.

2.2 Содержание отчёта

1. Номер работы и её название.

2. Задание для выполнения.
3. Код программы на языке Visual C++.
4. Описание входных и выходных данных.
5. Блок-схема программы.
6. Результат тестирования программы.

3.Вопросы к защите

1. Что в C++ называют структурой?
2. Каков формат объявления структуры в C++?
3. Для чего используются структуры?
4. Может ли отдельное поле структуры само быть структурой?
5. Допускается ли использование массивов структур?

4.Задание к выполнению

Составить блок-схему алгоритма и написать программу в соответствии с номером вашего варианта.

1. Распечатать фамилии детей детского сада, которые родились в определенном месяце, указать их возраст и группу.

2. Распечатать фамилии тех учеников, которые не получили ни одной тройки за последнюю четверть. В каких классах учатся эти ученики? Каков их средний балл?

3. Распечатать анкетные данные учеников, участвовавших в олимпиаде по информатике и заработавших не менее 30 баллов.

4. Среди работников данного предприятия найти тех, чья заработная плата за месяц ниже средней по предприятию и распечатать их список с указанием фамилии, стажа работы и должности.

5. На аптечном складе хранятся лекарства. Сведения о лекарствах содержатся в специальной ведомости: наименование лекарственного препарата; количество; цена; срок хранения (в месяцах). Выяснить, сколько стоят все препараты, хранящиеся на складе.

6. Вычислить средний балл учеников класса, если известны оценки каждого ученика по математике, русскому языку и физике. Распечатать список учеников, имеющих средний балл выше среднего в классе.

7. Из данного списка спортсменов распечатать сведения о тех из них, кто занимается плаванием. Указать возраст, сколько лет они занимаются плаванием.

8. Распечатать список учеников музыкальной школы, которые учатся играть на скрипке. Указать также, сколько лет они занимаются музыкой и принимали ли участие в каких – либо конкурсах.

9. Составить программу назначения стипендии студентам по результатам сессии, используя следующие правила:
    1) Если все оценки 5, назначается повышенная стипендия;
    2) Если все оценки 4 и 5 , назначается обычная стипендия;
    3) Если есть оценка 3, стипендия не назначается.

10. В таблице хранятся следующие данные об учениках: фамилия, имя, отчество, рост, масса. Вычислить средний рост учеников. Сколько учеников могут заниматься в баскетбольной секции, если рост баскетболиста должен быть больше 170 см.

11. При поступлении в университет лица, получившие оценку «неудовлетворительно» на первом экзамене, ко второму экзамену не допускаются. Считая фамилии абитуриентов и их оценки после первого экзамена исходными данными, составить список абитуриентов, допущенных ко второму экзамену.

12. Распечатать фамилии тех учеников класса, которые являются хорошистами и отличниками по итогам года. Также указать, насколько их средний балл отличается от среднего балла класса.
13. Создать структуры, определяющие положение точки в декартовой и полярной системах координат. Считая, что задан массив координат точек в декартовой системе координат, получить соответствующий массив координат заданных точек в полярной системе координат.

14. В таблице хранятся следующие данные о странах: название, столица, площадь территории, численность населения. Распечатать данные о тех странах, численность населения которых больше заданной величины.

15. Имеются данные об автомобилях: марка, номерной знак, год выпуска, стоимость. Вывести на экран информацию об автомобилях заданного года выпуска.

5.Пример. Распечатать список учеников, фамилии которых начинаются на букву В, с указанием даты их рождения.

5.1. Описание переменных

`stud` – массив структур типа "student";
`fam, grup, data` – поля структуры;
`N` – количество элементов массива;
`i` – индекс.

5.2.Текст программы

```
#include<iostream>//  библиотека ввода-вывода
#include<locale.h>// для переключения на русский язык
#include<conio.h>
usingnamespace std;  // использовать пространство имён std
void main()      //Главная функция
{   setlocale(LC_ALL,".1251"); // возможность переключения на
русский язык
int i,N;
    cout<<"Введите количество студентов:  ";
    cin>>N;
//Описание структуры «студент»
struct student { char fam[15];     //Фамилия
       int grup;          //№ группы
       char data[15];    //Дата рождения
                  };
   student stud[20];     //Массив переменных типа «студент»
for(i=0;i<N;i++)     //Заполнение массива
```

```
{    cout<<"студент"<<i+1;
cout<<"\nфамилия:   "); cin>>stud[i].fam;
cout<<"\nГруппа:   "); cin>>stud[i].grup;
cout<<"\nДата:   "); cin>>stud[i].data);
cout<<endl;
    }
for(i=0;i<N;i++)
if(stud[i].fam[0]=='B')                    //Вывод на экран
      {  cout<<"\nфамилия  ",stud[i].fam; //данных о
   cout<<"  Группа  ",stud[i].grup);     //студентах фамилии
которых
     cout<<"  Дата  ",stud[i].data;       //начинаются с буквы B.
}
   getch();
}
```

5.3.Тестирование программы

Результат работы программы может выглядеть следующим образом: