МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

КЫРГЫЗСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. И. Раззакова

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Кафедра «Автоматическое управление»

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА С++

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Для студентов направления 700200 «Управление в технических системах» всех форм обучения

Бишкек - 2014

«Рассмотрено»
На заседании кафедры
«Автоматическое управление»
Прот. №.4 от 16.12.2014 г.

«Одобрено»
Методическим советом
ФИТ
Прот. № 6 от 17.12.2014 г.

Составитель: Кудакеева Г.М.

Методические указания к выполнению лабораторных работ по дисциплине «Основы программирования на С++» для студентов направления **700200 «Управление в технических системах»** всех форм обучения / КГТУ им. И. Раззакова; Сост.: Кудакеева Г.М. / - Б.: ИЦ «Текник», 2014. - 28 с.

Излагается методика выполнения лабораторных работ, краткие теоретические сведения, примеры и задачи. Написаны в соответствии с учебной программой по дисциплине «Основы программирования на С++», утвержденный на заседании кафедры «Автоматическое управление».

Предназначены для студентов всех форм обучения.

Библиогр. 10

Рецензент к.т.н., доцент Михеева Н.И.

Содержание

Введение

Язык Си был создан в 1972 г. сотрудником фирмы Bell Laboratories в США Денисом Ритчи. По замыслу автора, язык Си должен был обладать противоречивыми свойствами. С одной стороны, это язык программирования высокого уровня, поддерживающий методику структурного программирования (подобно Паскалю). С другой стороны, этот язык должен обеспечивать возможность создавать такие системные программы, как компиляторы и операционные системы. До появления Си подобные программы писались исключительно на языках низкого уровня — Ассемблере, Автокоде. Первым системным программным продуктом, разработанным с помощью Си, стала операционная система UNIX. Из-за упомянутой выше двойственности свойств нередко в литературе язык Си называют языком среднего уровня. Стандарт Си был утвержден в 1983 г. Американским национальным институтом стандартов (ANSI) и получил название ANSI C. В начале 1980-х гг. в той же фирме Bell Laboratories ее сотрудником Бьерном Страуструпом было разработано расширение языка Си, предназначенное для объектно-ориентированного программирования. По сути дела, был создан новый язык, первоначально названный «Си с классами», а позднее (в 1983 г.) получивший название Си++ (Си-плюс-плюс).

C++ - это наиболее полный и совершенный продукт разработки программного обеспечения на сегодняшний день. Он обеспечивает высокий уровень скорости и удобства программирования, в то же время он предлагает широкий набор разнообразных форм проектирования, удобный практически для любого стиля программирования.

Самое существенное улучшение по сравнению с языком C (явившимся основой для создания C++) касается концепции объектно-ориентированного программирования.

Эти задания предназначены для закрепления теоретических сведений и в приобретении навыков по программированию.

Методические указания включают 5 лабораторных работ. Описание каждой лабораторной работы содержит краткие теоретические сведения, что позволяет варьировать объем заданий, индивидуализировать процесс обучения.

Лабораторная работа №1

Принципы создания программы на C++

Цель работы *- освоение основ программирования на языке C++.*

В данной работе рассматривается линейные или последовательные алгоритмы, т.е. такие алгоритмы, в которых действия выполняются друг за другом в той последовательности, в которой они написаны.

Краткие теоретические сведения

Технология создания программы

Чтобы создать готовую для использования компьютерную программу (неважно, простую или сложную), необходимы следующие действия:

- составление алгоритма;
- подготовка исходного текста программы;
- компиляция исходного текста;
- компоновка программы.

Исходный текст - это запись подробного алгоритма решения задачи на особом языке, удобном для восприятия человеком. Его называют языком программирования высокого уровня. Примеры таких языков: Basic, Pascal, C++, Java. Подготовка исходного текста программы - «ручная» работа, она выполняется программистом с помощью текстового редактора.

Программа, записанная на языке высокого уровня, понятна человеку, но не может быть непосредственно исполнена процессором компьютера. Чтобы процессор мог воспринимать программу и выполнять ее, программа предварительно должна быть переведена с языка высокого уровня на язык низкого уровня - машинный язык. Перевод исходного текста программы на машинный язык называют компиляцией программы. Такой перевод может быть выполнен автоматически, с помощью специальной компьютерной программы - компилятора.

Однако в результате компиляции еще не получается готовая к использованию программа, а получается всего лишь «полуфабрикат». Его необходимо объединить со специальными подпрограммами, которые «умеют» выполнять часто встречающиеся стандартные действия, например, вывод символов на экран, чтение символов, набираемых на клавиатуре, вычисление математических функций и тому подобное. Подпрограммы, предназначенные для выполнения таких стандартных действий, подготовлены другими программистами и помещены в так называемые библиотеки подпрограмм. Объединение вновь создаваемой программы с библиотечными подпрограммами называется компоновкой и выполняется автоматически с помощью специальной программы-компоновщика.

Для удобства программиста текстовый редактор, компилятор и компоновщик часто собраны в единую среду разработки программ и легко запускаются нажатием определенного сочетания клавиш или с помощью выбора мышью пунктов меню. Благодаря этому, на практике создание небольшой компьютерной программы не вызывает трудностей даже у новичка. Отличным примером среды разработки является **Microsoft Visual Studio 2006.**

Пример простейшей программы: Сложение двух чисел

Мы начинаем с рассмотрения простой программы, печатающей строку текста. Программа и результаты ее работы на экране показаны ниже

```
// Программа сложение 2 чисел
#include <iostream.h>
      main ()
      {
int a, b, sum;                                  //объявление
           cout << "Введите первое число\n";     //приглашение
           cin >> a;                                  //чтение целого
          cout << "Введите второе число\n";   //приглашение
cin >> b                                        //чтение целого
sum = a+ b;                                     //присваивание значения сумме
    cout << "Сумма равна " << sum<< endl;      //печать суммы
return 0;
```

Листинг программы:

Введите первое число

45

Введите второе число

72

Сумма равна 117

Эта программа иллюстрирует несколько важных свойств языка C++. Рассмотрим детально каждую строку программы.

Строка

// Программа сложения 2 чисел

начинается с символа //, показывающего, что остальная часть строки — это *комментарий.* Комментарии помогают другим людям читать и понимать вашу программу. Комментарии не вызывают никаких действий компьютера при выполнении программы. Они игнорируются компилятором C++ и не вызывают генерации каких-либо объектных кодов на машинном языке. Комментарий **Программа сложения 2 чисел** просто описывает цель программы. Комментарий, который начинается с //, называется *однострочным комментарием,* потому что комментарий заканчивается в конце текущей строки.

Строка

#include <iostream.h>

является *директивой препроцессора,* т.е. сообщением препроцессору C++. Строки, начинающиеся с #, обрабатываются препроцессором перед компиляцией программы. Данная строка дает указание препроцессору включить в программу содержание *головного файла потока ввода/вывода* **iostream.h.** Этот файл должен быть включен для всех программ, которые выводят данные на экран или вводят данные с клавиатуры, используя принятый в C++ стиль, основанный на понятии потока ввода-вывода.

Строка

main()

является частью каждой программы на C++. Круглые скобки после **main** показывают, что **main** — это программный блок, называемый *функцией.*

Программа на С++ содержит одну или более функций, одна из которых должна быть **main.**

Левая фигурная скобка { должна начинать *тело* каждой функции. Соответствующая *правая фигурная скобка* должна заканчивать каждую функцию.

Строка

int a, b, sum; *// называется объявлением.*

Символы **a, b и sum** являются именами *переменных.* ***Переменная*** — это область в памяти компьютера, где может храниться некоторое значение для использования его в программе. Данное объявление определяет, что переменные a, b и sum имеют тип данных int; это значит, что эти переменные всегда будут содержать *целые значения,* т.е. целые числа, такие как 7, -11, 0, 31914. Все переменные должны объявляться с указанием имени и типа данных прежде, чем они могут быть использованы в программе.

Переменные должны объявляться с указанием имени и типа данных в любом месте программы, но раньше, чем они будут использованы в программе. Для объявления переменных, используются следующие типы данных:

Тип	Размер в байтах	Диапазон значений
char	1	От –128 до 127 – символьный тип
short	2	От –32768 до 32767 короткое целое число
int	4	От –2147483648 до 2147483647 – целое число
long	4	Совпадает с int – целое число
float	4	От 1,2E-38 до 3,4E+38 – вещественное число одинарной точности
double	8	От 2,2E-308 до 1,8E+308 -вещественное число двойной точности

Строка

cout « " Введите первое число\n ";

печатает на экране буквенное сообщение «**Введите первое число»** и

позиционирует курсор на начало следующей строки. Это сообщение называется *приглашением,* потому что оно предлагает пользователю выполнить некоторое действие. О предыдущем операторе можно сказать так: **«cout** *получает* символьную строку **"Введите первое целое число\n".** ***Оператор***

cin >> a; //чтение целого

использует объект входного потока **cin** и операцию взять из потока ">>", чтобы получить от пользователя значение. Объект **cin** забирает вводимую информацию из стандартного потока ввода, которым обычно является клавиатура. О предыдущем операторе можно сказать так, «cin *дает* значение первого целого числа».

Когда компьютер выполняет предыдущий оператор, он ждет от пользователя ввода значения переменной **a.** В ответ пользователь набирает на клавиатуре целое число, и затем нажимает *клавишу возврата* — **return** (называемую иногда *клавишей ввода* — **enter),** чтобы послать это число в компьютер. Компьютер затем присваивает это число, или *значение,* переменной **a.** Любое последующее обращение в программе к **a** будет использовать это значение.

Объекты потоков **cout** и **cin** вызывают взаимодействие между пользователем и компьютером. Поскольку это взаимодействие напоминает диалог, часто говорят о *диалоговом расчете или интерактивном расчете.*

Оператор присваивания

sum = a + b;//присваивание значения сумме

рассчитывает сумму переменных **a** и **b** и присваивает результат переменной **sum,** используя *операцию присваивания* =. Оператор читается так, **«sum** *получает* значение, равное **a** + **b».** Оператор присваивания используется в большинстве расчетов. Операция = и операция + называются *бинарными операциями,* потому что каждая из них имеет по два *операнда.* В случае операции + этими операндами являются **a** и **b.** В случае операции = двумя операндами являются **sum,** и значение выражения **a + b** является командой компьютеру напечатать на экране строку символов, заключенную в кавычки.

Оператор

cout << "Сумма равна " << sum << endl; //печать суммы

Полная строка, включающая **cout,** операцию << - двойной знак меньше, строку " Сумма равна \n" и точка с запятой (;), называется *оператором.* Каждый оператор заканчиваться точкой с запятой (известной также как *признак конца оператора).* Все вводы и выводы в С++ выполняются над *потоками* символов. Таким образом, когда выполняется предыдущий оператор, он посылает поток символов **Сумма равна** объекту *стандартный поток вывода* **cout,** который обычно «связан» с экраном.

Операция двойной знак "<<" - меньше называется *операцией поместить в поток.* При выполнении этой программы значение справа от оператора, правый *операнд,* помещается в поток вывода. Символы правого операнда обычно выводятся в точности так, как они выглядят между двойными кавычками. Заметим, однако, что символы \n не выводятся на экране. Обратный слэш (\) называется *знаком перехода* или *escape-символом (эскейп).* Он свидетельствует о том, что должен выводиться «специальный» символ. Когда обратный слэш встречается в цепочке символов, следующий символ комбинируется с обратным слэшем и формирует *управляющую последовательность (escape-последовательность).* Управляющая последовательность \n означает *новую строку.* Она вызывает перемещение курсора (т.е. индикатора текущей позиции на экране) к началу следующей строки на экране.

Строка

return 0; //показывает, что программа успешно окончена

включается в конце каждой функции **main.** Ключевое слово С++ **return** — один из нескольких способов *выхода из функции.* Когда оператор **return** используется в конце **main,** как в этой программе, значение 0 свидетельствует о том, что программа завершена успешно. Правая фигурная скобка } означает окончание **main.**

В программах могут выполняться арифметические вычисления. Язык С++ содержит арифметические операции сложение (+), вычитание (-), деление (/), умножение (*) и вычисления остатка от деления (%). Приоритет выполнения

операций такой же, как и в обычных алгебраических выражениях, т.е. сначала выполняются умножение, деление и вычисление остатка от деления, а затем сложение и вычитание. Приоритет можно изменить, как и в обычной алгебре, с помощью круглых скобок.

Арифметические операции используются как обычно, кроме двух особенностей:

1. целочисленное деление дает целый результат, т.е. если мы делим целое число на целое результат будет целым; например, выражение 7/4 равно 1, а выражение 17/5 равно 3. Заметим, что любая десятичная часть при целочисленном делении просто отбрасывается (т.е. усекается) — округление не производится. Если мы хотим получить в результате число с дробной частью, то после одного из чисел достаточно поставить знак точку; например, записать 7/4. и получить результат 1.75 или результат от 17./5 будет равен 3.4.

2. операция вычисления остатка % используется только для целых чисел, для вещественных чисел используется функция fmod, о которой вы узнаете ниже. Выражение x % y дает остаток от деления x на y. Таким образом, 7 % 4 равно 3; 17 % 5 равно 2. Эта операция используется для определения кратности, чётности, нечётности.

Для использования различных математических функций следует подключить библиотеку встроенных математических функций #include <math.h>

Функции библиотеки “math.h”	
Функция	Действие
sqrt(x)	$\sqrt{x}$
log(x)	ln x
pow(x,y)	x^y
fmod(x,y)	остаток от деления “x” на “y” (для вещественных чисел)
fabs(x)	|x|
log10(x)	lgx
ceil(x)	округляет “x” до ближайшего целого, не меньшего“x”,

	ceil(9.2)=10.0
floor(x)	округляет “x” до ближайшего целого, не превышающего “x”, floor(9.2)=9.0
exp(x)	e^x
sin(x), cos(x), tan(x), atan(x)	тригонометрические функции sinx, cosx, tgx, arctgx

Результаты выполнения заданий этой и в дальнейшем остальных лабораторных работ оформите в виде отчета в следующем порядке:

Титульный лист установленного образца

Разделы отчета:

Условие задачи:

Исходные данные:

Блок-схема:

Код программы:

Результаты вычислений:

Выводы

Литература

Задания для самостоятельной работы

Необходимо выполнить нижеследующие задания для самостоятельной работы на компьютере.

1. Даны два действительных числа a и b. Получить их сумму, разность и произведение.
2. Дана длина ребра куба. Найти объем куба и площадь его боковой поверхности.
3. Даны катеты прямоугольного треугольника. Найти его гипотенузу и площадь.
4. Определить время падения камня на поверхность земли с высоты h.
5. Дана сторона равностороннего треугольника. Найти площадь этого треугольника.

6. Дан радиус окружности. Вычислить длину окружности и площадь круга.

7. Найти площадь равнобочной трапеции с основаниями a и b и углом α при большем основании a. (Примечание: углы компилятор принимает в радианах).

8. Известны длины трех сторон треугольника. Вычислить площадь треугольника.

9. Напишите программу, которая просит пользователя ввести два числа, получает числа от пользователя и затем печатает сумму, произведение, разность и частное этих чисел.

10. Отобразите модель шахматной доски восемью операторами вывода и затем отобразите ту же модель наименьшим возможным количеством операторов вывода:

```
* * * * * * * *
 * * * * * * * *  * * * * * * * *
 * * * * * * * *  * * * * * * * *
 * * * * * * * *  * * * * * * * *
* * * * * * * *
```

Лабораторная работа № 2

Операторы условной и безусловной передачи управления

Цель работы – *изучение структур разветвления.*

В данной работе используются разветвляющиеся алгоритмы.

Краткие теоретические сведения

Для реализации разветвляющихся алгоритмов используется условный оператор "if".

Оператор "if" имеет две формы записи:

1) if (<условие>) <оператор>;
2) if (<условие>) <оператор1>; else <оператор2>;

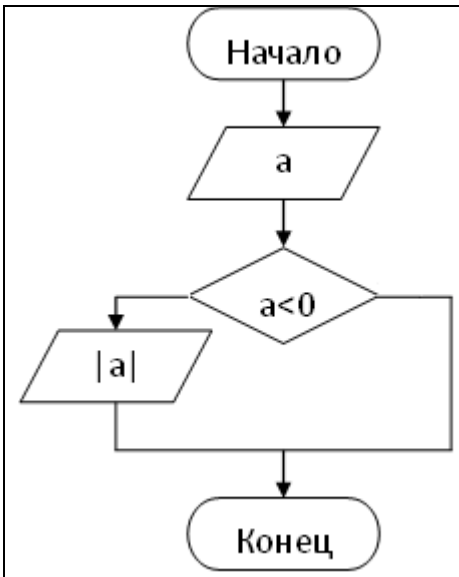
где условие – логическое выражение, переменная или константа.

Рассмотрим первый вид if (<условие>) <оператор>;

Структура if (ЕСЛИ) называется *структурой с единственным выбором,* поскольку она выбирает или игнорирует единственное действие.

Структура выбора if выполняет некоторое действие (обработку), если проверяемое условие истинно, или пропускает его, если условие ложно.

Пример: Дано вещественное число. Если оно отрицательно, то вывести его абсолютное значение.




```
#include <iostream>
#include <cmath>
using namespace std;
int main() {
    double a;
    cin>>a;
    if (a<0)
      cout<<fabs(a)<<endl;
    return 0;
}
```

Теперь рассмотрим второй вид if (<условие>) <оператор1>; else <оператор2>;

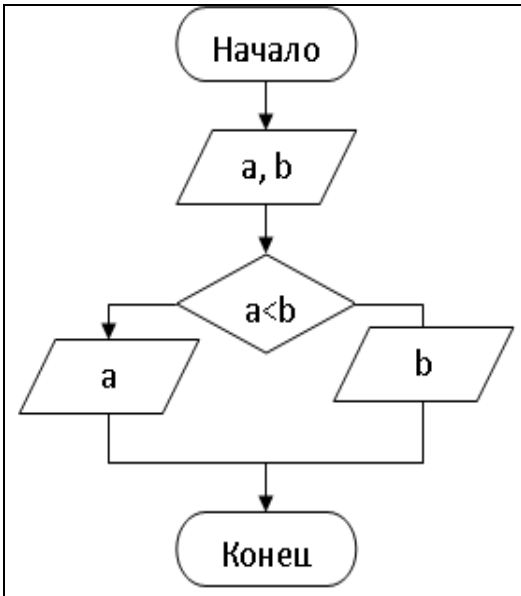
Таким образом, если в операторе **if** требуется, чтобы в зависимости от значения условия выполнялся не один оператор, а несколько, то оператор **if** следует записывать в следующей форме:

if (<условие>) <оператор1>; else <оператор2>;

Структура if/else (ЕСЛИ-ИНАЧЕ) называется *структурой с двойным выбором,* так как она осуществляет выбор между двумя различными действиями.

Структура выбора if/else выполняет одно действие, если условие истинно, и выполняет другое действие, если оно ложно.

Пример: Даны два целых числа. Вывести наименьшее из них. Алгоритм и программа решения данной задачи представлены ниже в таблице.




```
#include<iostream.h>
#include<math.h>
void main()
{
double x,y;
cout<<"input x"<<endl;
cin>>x;
if(x>=0)
        y=sqrt(x+2);
        else
                y=(x*x+3)/2;
        cout<<"your result is
"<<y<<endl;
}
```

Большинство операторов языка С++ в соответствии с синтаксисом могут выполнить только одно действие. В случае, если по условию задачи требуется

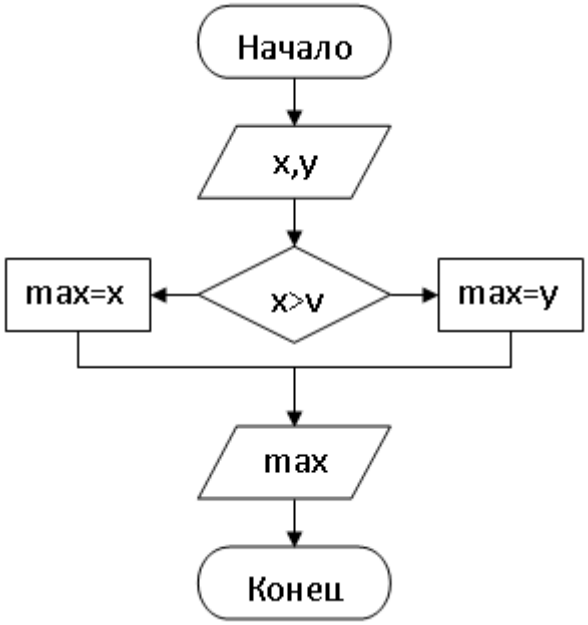
выполнить последовательно несколько действий, то последовательность соответствующих операторов должны быть заключены в фигурные скобки {}. Операторы, идущие после оператора if и заключённые в фигурные скобки, называются ”телом” оператора if.

Имеется более короткий способ записи условного оператора:

(<условие>)?<оператор1>:<оператор2>;

Проверяется условие: если оно истинно, то выполняется “оператор1”, если оно ложно, то выполняется “оператор2”. Применение данного способа записи приводит к получению более компактного машинного кода.

Пример: Даны 2 действительных числа. Найти наибольшее из этих 2-х чисел. Алгоритм и программа решения данной задачи представлены ниже в таблице.




```
#include <iostream.h>
int main() {
    double x, y, max;
    cin>>x>>y;
    max=(x>y)?x:y;
    cout<<”max=”<<max<<endl;
    return 0;
}
```

Для записи условия используют логические операции: >(больше), <(меньше), >=(больше или равно), <=(меньше или равно), = =(равно), !=(не равно) и логические операторы && (логическое и), || (логическое или), ! (отрицание). Условие может принимать 2 значения: true(истина) или false(ложь).

Для определения истинности сложного логического условия (выражения), записанного с помощью логических операций и операторов, применяется таблица истинности, где 1 – истина (true), а 0 – ложь (false).

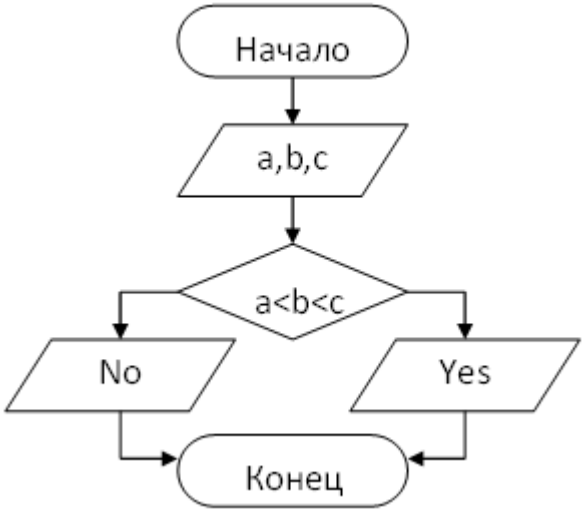
Таблица истинности				
x	y	x&&y	x||y	!x
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Логический оператор “&&” (и) даёт истинное значение логического выражения только в том случае, когда оба операнда этого оператора являются истинными. Например: if ((x==5)&&(y==5)) – это выражение буден истинным только в том случае, когда x=5 и y=5.

Логический оператор “||” (или) даёт истинное значение логического выражения в том случае, когда хотя бы один из операндов этого оператора является истинным. Например: if ((x==5)||(y==5)) – это выражение буден истинным в том случае, когда или x=5 или y=5.

Логические операции так же как и арифметические операции имеют приоритет выполнения. Логические операции (>, <, <=, >=) имеют больший приоритет, чем логические операторы (! , || , &&).

Пример использования логических операторов: Даны 3 действительных числа. Проверить правильно ли выполняется следующее соотношение: a<b<c; Алгоритм и программа решения данной задачи представлены ниже в таблице.

<table>
<tr>
<td> 
</td>
<td>#include <iostream.h>
int main()
{ double a, b, c;
cin>>a>>b>>c;
if ((a<b)&&(b<c))
cout<<”Yes”;
else
cout<<”No”;

return 0;}</td>
</tr>
</table>

Оператор множественного выбора “switch”

Мы рассмотрели структуру с единственным выбором “if” и структуру с двойным выбором “if/else”. Но порой алгоритм может содержать ряд альтернативных решений, причем некоторую переменную надо проверять отдельно для каждого постоянного целого значения, которое она может принимать; в зависимости от результатов этой проверки должны выполняться различные действия. Для решения этой проблемы предусмотрен оператор “switch”, который позволяет рассматривать сразу несколько условий.

switch (выражение)
{
case значение_1: Оператор_1; break;
case значение_2: Оператор_2; break;
case значение_3: Оператор_3; break;
...
case значение_n: Оператор_n; break;
default: Оператор; break;
}

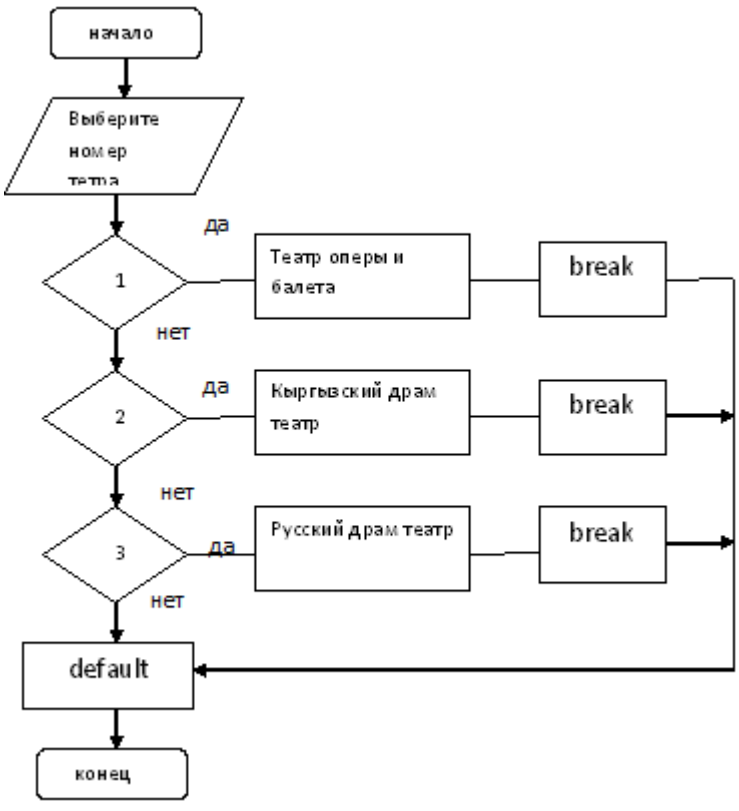
Выражение, заданное в скобках оператора “switch” сравнивается со значениями, указанными за операторами “case”. В случае совпадения значений

выражения, выполняется оператор в строке соответствующего оператора “case”. Будут выполняться все строки программы после выбранного оператора “case” до тех пор, пока не закончится тело блока оператора “switch” или не повстречается оператор “break”. Выполнение оператора “break” приводит к выходу из оператора “switch”. Если “break” отсутствует, то управление передаётся следующему оператору “case” или оператору “default”. Если ни одно из значений операторов “case” не совпадает с выражением, то выполняются строки программы, стоящие после оператора “default”. Наличие оператора “default” не обязательно. В случае его отсутствия и несовпадения выражения ни с одним из значений, будет выполнен оператор стоящий после блока оператора “switch”.

Примечание: данная конструкция используется только для типа int или char .

Пример: Составить список репертуаров всех театров Бишкека: создать простейшее меню, позволяющие выбрать № театра, название спектакля, время начала и конца спектакля.

Алгоритм и программа решения данной задачи представлены ниже в таблице.




```
#include <iostream.h>
 void main()
 {
int nomer;
cout<<"Viberite Kino \n 1.Tor \n 2.Mstiteli \n 3.KT\n 4.kurmanjan datka\n 5.Velikii  Getsby";
cin>>nomer;
switch (nomer)
{
    case 1: cout<<"250 som,nachalo 18:00, konets 20:00\n";break;
    case 2:  cout<<"200
```

	som,nachalo 19:30, konets 21:00\n";break; case 3: cout<<"300 som,nachalo 20:45, konets 22:00\n";break; case 4: cout<<"150 som,nachalo 15:00, konets 17:00\n";break; case 5: cout<<"190 som,nachalo 15:00, konets 17:00\n";break; default: cout<<"takogo filma net \n"; } }

Задания для самостоятельной работы

1. Если x>=0 $y = \sqrt{x + 2}$, иначе $y = (x^2 + 3)/2;$
2. Написать программу, которая определяет, является ли введенное пользователем предложение палиндромом (перевертышем) «кит на море не романтик».
3. Даны три действительных числа. Возвести в квадрат те из них значения, которых неотрицательны.
4. Напишите программу, вводящую три целых числа и печатающую Yes в том случае, если среди введенных чисел есть одинаковые, и No - иначе.
5. Дано целое трехзначное число. Определить является ли первая цифра этого числа больше, чем последняя.
6. На билете пять цифр - номер билета. Является ли данный билет счастливым? Счастливым считается билет, если сумма трех левых цифр равна сумме трех правых цифр.

7. Даны действительные числа a, b, c. Найти корни квадратного уравнения $ax^2+bx+c=0$, в противном случае ответом должно служить сообщение, что корней нет.

8. Дано четырехзначное число. Найти сумму цифр четырехзначного числа.

9. Дано число. Определить, является ли данное число четным или нечетным.

10. В старояпонском календаре был принят двенадцатилетний цикл. Годы внутри цикла носили названия животных: крысы, коровы, тигра, зайца, дракона, змеи, лошади, овцы, обезьяны, петуха, собаки и свиньи. Написать программу, которая позволяет ввести номер года и печатает его название по старояпонскому календарю. Справка: 1996 г. — год крысы — начало очередного цикла. (Поскольку цикл является двенадцатилетним, поставим название года в соответствие остатку от деления номера этого года на 12).

Лабораторная работа № 3

Циклические алгоритмы и программы

Цель работы – *изучение операторов цикла языка С++.*

Краткие теоретические сведения

Одним из самых замечательных достоинств компьютера является то, что он способен автоматически повторять однообразные действия, очень быстро и без ошибок.

Многократное исполнение одной и той же группы инструкций называется в программировании циклом (англ. C1гc1e - круг).

Любой цикл состоит из:

- тела цикла, т.е. операторов, повторяющихся несколько раз;
- начальных значений;
- модификации (изменения) параметра цикла;
- проверки условия продолжения цикла.

Один проход цикла называется итерацией. Итерация – это очередное выполнение тело цикла, с новым значением параметра цикла. В С++ для удобства существует три типа операторов цикла. Это - оператор цикла с предусловием while, оператор цикла с постусловием do/while и оператор цикла for. Поставленную задачу можно решить, используя любой из них. Рассмотрим их подробнее.

Оператор цикла с предусловием "while"

Во многих случаях для организации цикла, вместо оператора "for", бывает удобнее использовать оператор "while" (англ. while - пока). Этот оператор цикла записывается в виде:

while (< условие>) { <операторы>};

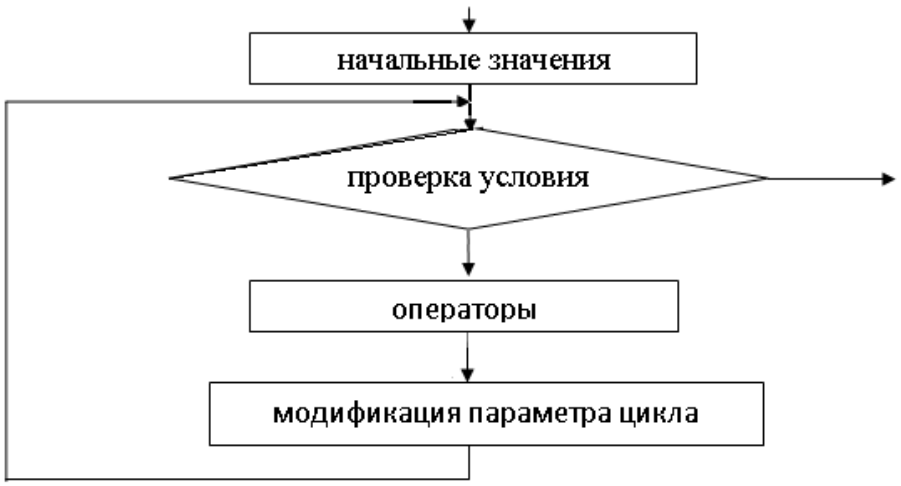
Работает оператор "while" так:

- проверяется указанное в скобках условие;
- если оно удовлетворяется, выполняется «тело» цикла - одиночный или составной оператор, и опять происходит переход к проверке условия;
- если условие не удовлетворяется - оператор цикла завершает свою работу и управление передается следующему за телом цикла оператору программы.

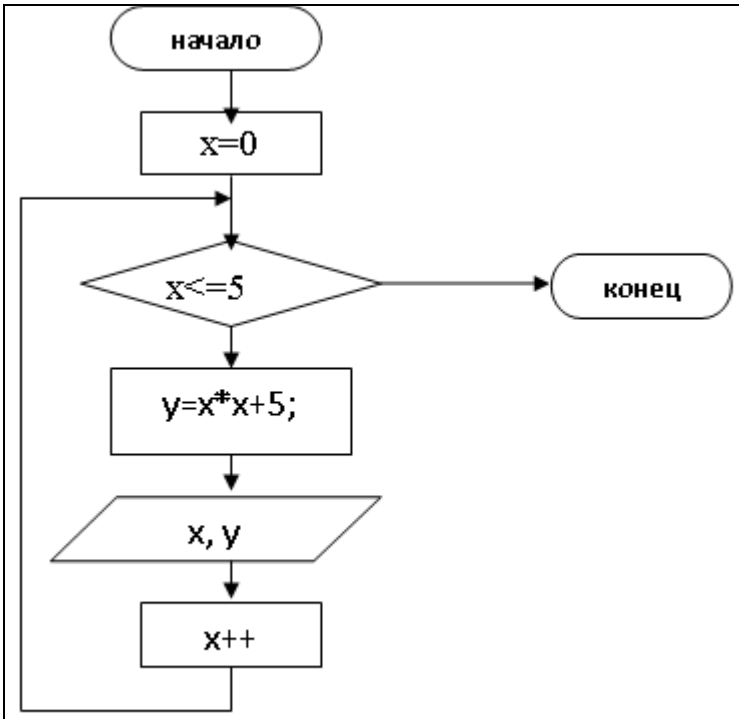
Кратко оператор цикла "while" можно описать словами:

пока удовлетворяется условие, повторяется выполнение «тела» цикла.

Соответствующий алгоритм показан ниже**.**



Пример: Вычислить значения функции $y=x^2+5$, если $x \in [0,5]$ с шагом 1. Алгоритм и программа решения данной задачи представлены ниже в таблице.



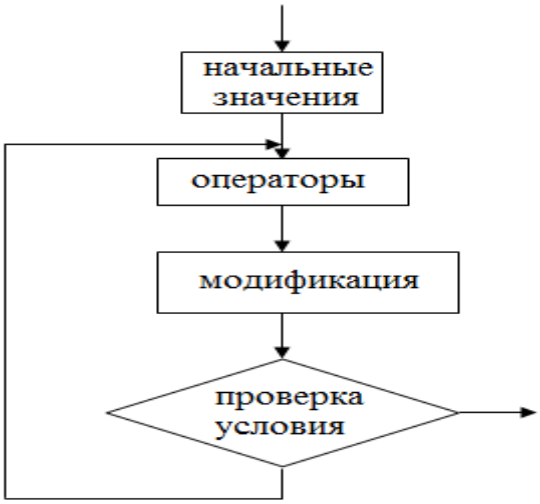

```
#include <iostream.h>
int main() {
  int  x, y;
  x=0;
  while (x<=5)
{
y=x*x+5;
cout<<”x=”<<x<<”y=”<<y
<<endl;
        x++;
}

 return 0; }
```

Оператор цикла с постусловием “do/while”

Структура повторения **do/while** похожа на структуру **while.** В структуре **while** условие продолжения циклов проверяется в начале цикла, до того, как выполняется тело цикла. В структуре **do/while** проверка условия продолжения циклов производится *после* того, как тело цикла выполнено, следовательно, тело цикла будет выполнено по крайней мере один раз. Когда **do/while** завершается, выполнение программы продолжается с оператора, следующего за предложением **while.** Отметим, что в структуре **do/while** нет необходимости использовать фигурные скобки, если тело состоит только из одного оператора. Но фигурные скобки обычно все же ставят, чтобы избежать путаницы между структурами **while** и **do/while.**

Соответствующий алгоритм показан ниже**.**



Форма записи оператора цикла с постусловием “do/while”:

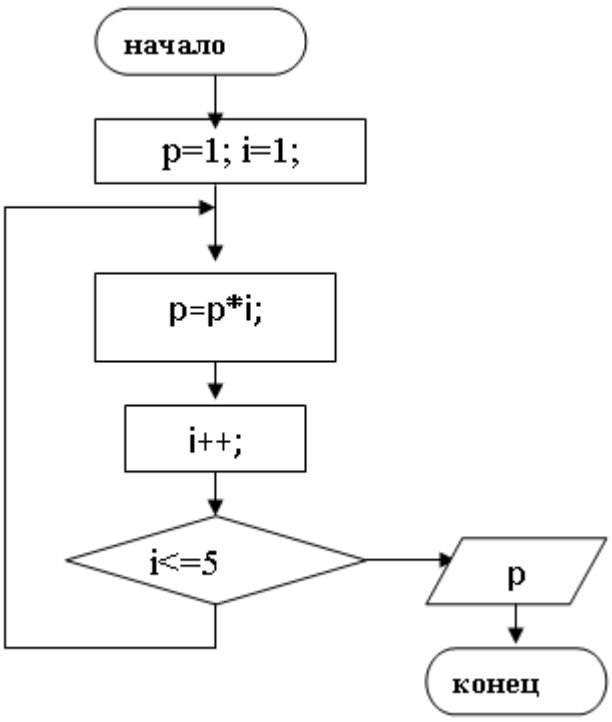
do

{<операторы>}

while(<условие>);

При использовании **оператора** условие проверяется после выполнения тела цикла, это гарантирует выполнение цикла хотя бы один раз, даже если условие всегда ложно.

Пример: Вычислить произведение целых чисел от 1 до 5. Алгоритм и программа решения данной задачи представлены ниже в таблице.



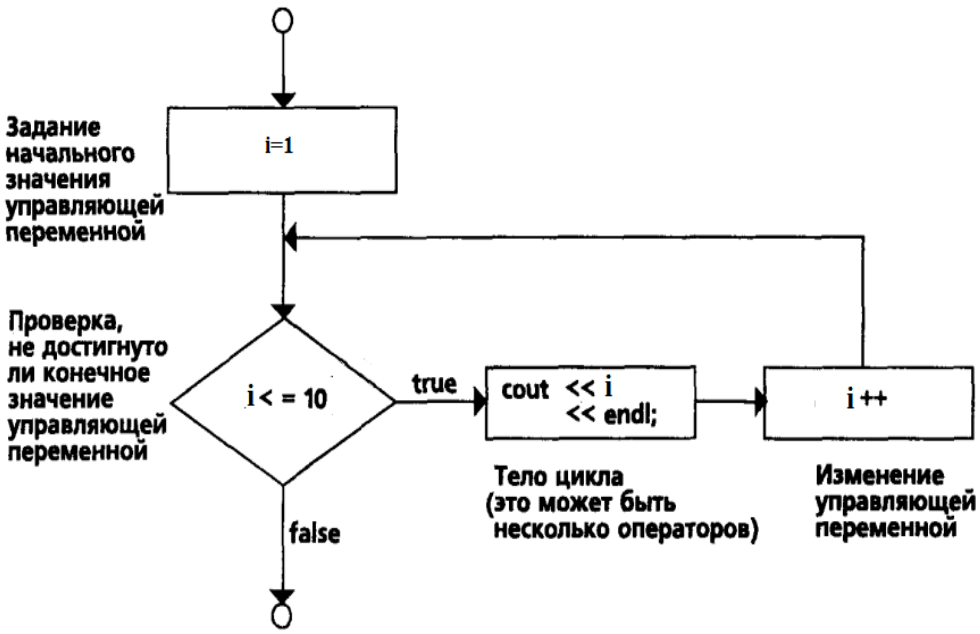

```
#include <iostream.h>
int main() {
  int p, i;
  p=1; i=1;
{
   p*=i;
        i++;
  }   while (i<=5);
  cout<<p<<endl;
 return 0;
}
```

Цикл с параметром (for)

Оператор for имеет три главные части:

- Инициализация – место, где обычно находится оператор присваивания, используемый для установки начального значения переменной цикла;
- Место, где находится выражение, определяющее условие работы цикла;
- Увеличение – место, где определяется характер изменения переменной цикла на каждой итерации.

Соответствующий алгоритм работы цикла ***for*** показан ниже**.**



1) Присваивается начальное значение счётчику цикла, т.е инициализируется счётчик цикла.

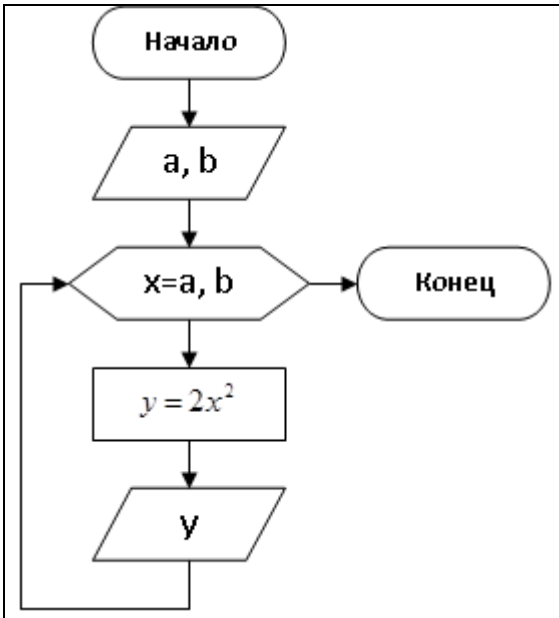
2) Выполняется проверка значения счётчика. Если результат проверки равен значению “true”, то сначала выполняется тело цикла, а за тем операция над счётчиком (приращение, уменьшение и т.д.).

Форма записи оператора цикла

for (<инициализация_счётчика>;<проверка>;<операция_над_счётчиком>) операторы;

Пример:

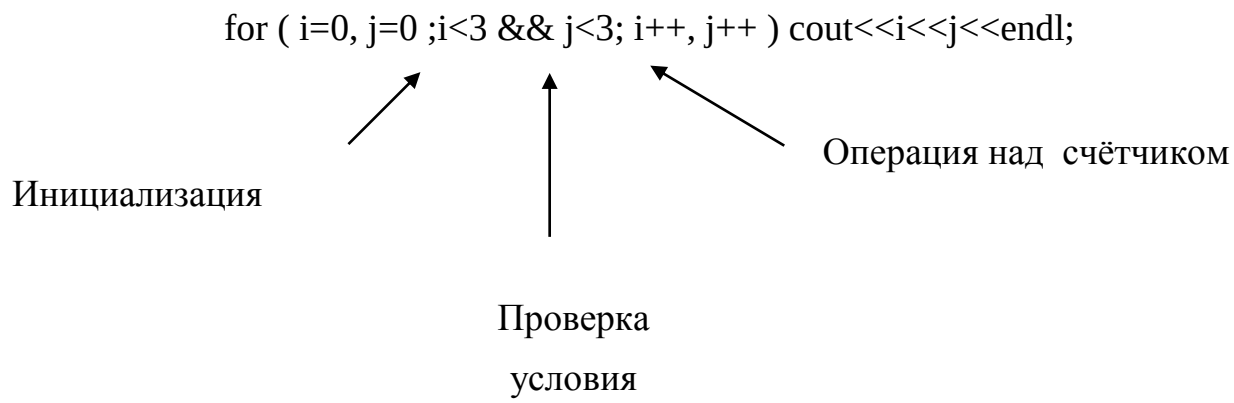
Вычислить значение функции $y = 2x^2$ для всех значений “x” в диапазоне x∈ [a;b], где a- нижняя граница диапазона, b- верхняя граница диапазона. Алгоритм и программа решения данной задачи представлены ниже в таблице.




```
#include <iostream.h>
#include <math.h>
int main() {
  int a,b,x;
  cin>>a>>b;
  double y;
  for (x=a; x<=b; x++)
{
    y=2 * pow ( x, 2);
    cout<<”y=”<<y<<endl;
  }
  return 0;
}
```

Множественная инициализация счётчика цикла for

Синтаксис оператора цикла for позволяет инициализировать несколько переменных счётчиков, проверять сложное условие продолжения цикла и последовательно выполнять несколько операций над счётчиками цикла. Если присваиваются значения нескольким счётчикам или выполняются операции с несколькими счётчиками, то они записываются последовательно, и разделяются запятыми.



Результатом этого фрагмента программы будет вывод чисел в виде:

00

11

22

Вложенные циклы "for"

Цикл, организованный внутри тела другого цикла, называется вложенным циклом. В этом случае внутренний цикл полностью выполняется на каждой итерации (шаге) внешнего цикла.

```
for ( i=0; i < 2; i++)
  for ( j=0; j < 2; j++)
    cout << i << j << endl;
```

Результатом этого фрагмента программы будет вывод чисел в виде:

00

01

10

11

20

21

Задания для самостоятельной работы

1. Найти сумму и произведение последовательности 10 целых чисел от 1 до 10.

2. Даны натуральное n и действительное x. Вычислить $\sin x + \sin^2 x + \ldots + \sin^n x$.

3. Напишите программу, которая читает пять чисел (каждое между 1 и 30). Для каждого просчитанного числа ваша программа должна напечатать строку, содержащую соответствующее число смежных звездочек. Например, если ваша программа прочла число 7, она должна напечатать *******.

4. Напишите программу, которая напечатает следующий ромб. Вы можете использовать операторы вывода, которые печатают или одну звездочку (*), или один пробел. Максимально используйте повторение (с вложенными структурами for) и минимизируйте число операторов вывода.

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

5. Дано натуральное n. Вычислить:

$$\frac{tgx}{2}+\frac{tgx^2}{3}+\frac{tgx^3}{4}+\frac{tgx^4}{5}+\frac{tgx^5}{6}+\ldots\frac{tgx^n}{n+1};$$

6. Напишите программу, которая печатает следующие трафареты один под другим. Используйте цикл for для генерации трафаретов. Все звездочки (*) должны печататься одним оператором вида cout <<'*'; (в результате звездочки будут печататься рядами). Подсказка: два последних трафарета требуют, чтобы каждая строка начиналась с соответствующего числа пробелов. Задача повышенной сложности: объедините ваши коды для решения четырех отдельных задач в единую программу, которая печатала бы все четыре трафарета рядом с помощью вложенных циклов **for.**

(A)	(B)	(C)	(D)
*	**********	**********	*
**	*********	*********	**
***	********	********	***
****	*******	*******	****
*****	******	******	*****
******	*****	*****	******
*******	****	****	*******
********	***	***	********
*********	**	**	*********
**********	*	*	**********

7. Напишите программу, которая считает и печатает произведение нечетных целых от 1 до 15.

8. Вычислить сумму натуральных четных чисел, не превышающих N.

9. Из 5-ти чисел найти кол-во четных и нечетных чисел и найти соответственно их суммы.

10. Определить количество простых чисел в интервале от N до M, где N и M натуральные числа.

Лабораторная работа №4

Работа с одномерными массивами

Цель работы – *построение схем алгоритмов с использованием массивов.*

Краткие теоретические сведения

Массив – это совокупность нескольких данных одного типа, которые расположены в подряд идущих ячейках памяти. Элементы массива занимают один непрерывный участок памяти компьютера и располагаются последовательно друг за другом. Нумерация элементов в массиве начинается с 0 и заканчивается n-1 элементом, где n-количество элементов массива.

C++ в отличие от других языков не запрещает запись в любой участок памяти, поэтому программисту самому надо следить за тем, чтобы не произошел выход за границы массива, так как это может привести к прекращению работы операционной системы.

Для того, чтобы объявить массив, необходимо:

1. Указать тип данных информации;

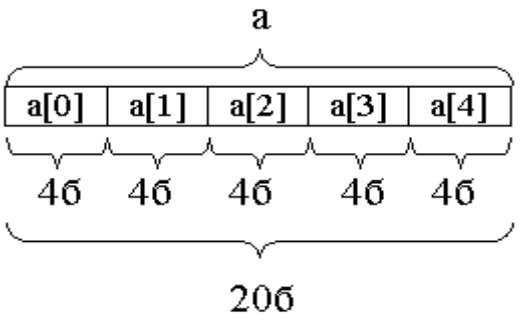
2. Указать имя массива (имя массива может быть любое);
3. В квадратных скобках указать количество элементов в массиве.

Синтаксис описания одномерного массива:

<Тип данных> <имя массива> [<количество элементов>];

Например: int mass [5];

Массив из 5 целочисленных элементов, имя массива – mass, занимает непрерывный участок памяти размером 20 байтов, нумерация начинается с 0 и заканчивается 4.



Одновременно с описанием массива можно инициализировать его элементы. Те значения, в которых инициализируется каждый элемент массива, перечисляются в фигурных скобках через запятую, количество элементов инициализации не может быть больше размерности массива.

1) int mass[5]={ 5, 4, 2, 8,1 };

Во втором случае компилятор сам определяет количество элементов массива по списку значений и выделяет соответствующее количество памяти.

2) int mass []={ 5, 4, 2, 8, 1};

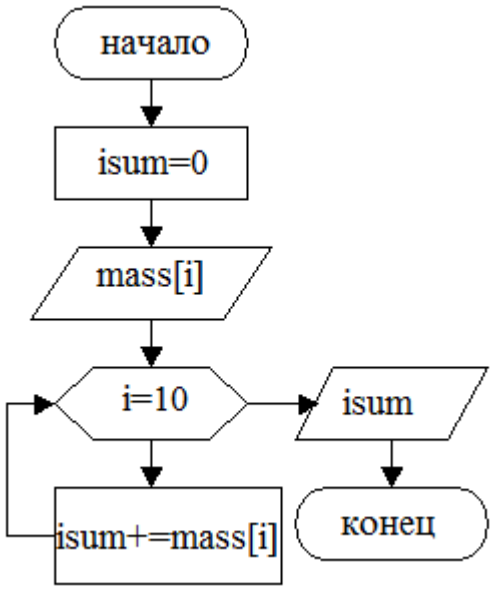
Также можно проинициализировать какие либо значения, какого- либо массива следующим образом: mass [0]=5; mass [1]=4; и т.д.

Для работы с массивами используются операторы цикла.

```
for ( i=0; i<n; i++)    cin>>a[i];        //Ввод массива
for ( i=0; i<n; i++)    cout<<a[i];       //Вывод массива
```

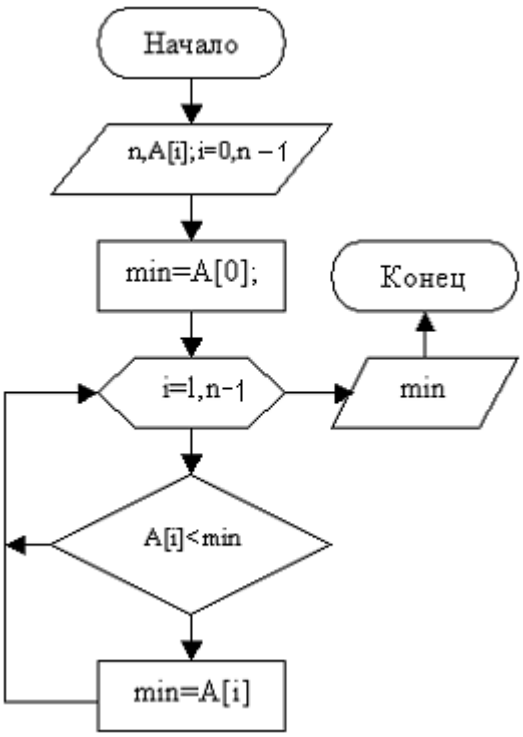
Стандартные алгоритмы обработки одномерных массивов

1. Поиск суммы элементов массива:




```
#include <iostream.h>
void main()
{   int mass[10]={0};
    int isum=0;
    for (i=0;i<10;i++)
cin>>mass[i];
for (i=0;i<5;i++) {
isum+=A[i];  }
cout<<”otvet=”<<isum<<endl;
  }
```

2. Поиск минимального элемента массива:




```
#include <iostream.h>
int main() {
  int A[100], n, i, min;
  cin>>n;
  for ( i=0; i<n; i++)
    cin>>A[i];
  min=A[0];
  for (i=1; i<n; i++)
    if (A[i]<min)
      min=A[i];
  cout<<min<<endl;
  return 0;
}
```

Задания для самостоятельной работы

1. Дан массив чисел. Определить, сколько в нем пар одинаковых соседних элементов.
2. Дан массив чисел. Найти сумму чисел кратных 3, и подсчитать их количество.
3. Найти произведение элементов массива с нечеткими индексами, размерностью 20, тип данных int;
4. Дан массив чисел тип данных int, размерность 24, заменить все числа 6 на 0, подсчитать количество замен.
5. Дан массив чисел размерностью 20, типа int, из него сформировать массив который будет заполнен только теми значениями первого массива, у которых индексы четные.
6. Дан массив чисел размерностью 20, тип данных int, из этого массива сформировать другой массив, размер которого два раза меньше.
7. Дан массив чисел тип данных float, размерность 12. Подсчитать сколько раз встретилось число 3,1.

Лабораторная работа № 5

Работа с двумерными массивами

Цель работы – *разработка программ с использованием двумерных массивов.*

Краткие теоретические сведения

В предыдущей лабораторной работе мы рассматривали одномерные массивы, но существуют двумерные массивы. Двумерный массив – это матрица размером nxm. Например, двумерный массив int mass[4][4] можно представить как 4 массива типа int по 5 элементов в каждом. Представим его в виде матрицы размером 4x4:

	0-ой столбец	1-ый столбец	2-ой столбец	3-ий столбец	4-ый столбец
0-ая строка	v[0][0]	v[0][1]	v[0][2]	v[0][3]	v[0][4]
1-ая строка	v[1][0]	v[1][1]	v[1][2]	v[1][3]	v[1][4]
2-ая строка	v[2][0]	v[2][1]	v[2][2]	v[2][3]	v[2][4]
3-ая строка	v[3][0]	v[3][1]	v[3][2]	v[3][3]	v[3][4]

Нумерация строк и столбцов начинается с 0 и заканчивается n-1. Первый индекс – номер строки, второй индекс - номер столбца.

Объявление двумерного массива:

<тип элементов> <имя > [количество строк][количество
массива столбцов];

Например: int mass [3][3];

Вместе с объявлением можно проинициализировать массив, инициализация двумерного массива выполняется следующим образом:

int mass [3][3]={{4,5,7},{3,7,6},{2,1,8}};

Элементы каждой отдельной строки заключены в отдельные фигурные скобки, элементы отделяются друг от друга запятой. Обращаться к каждому элементу массива можно с помощью определения номера строки и столбца

Ввод двумерного массива с клавиатуры:

```
for ( i=0; i<3; i++)
  for ( j=0; j<3; j++)
    cin>> mass [i][j];
```

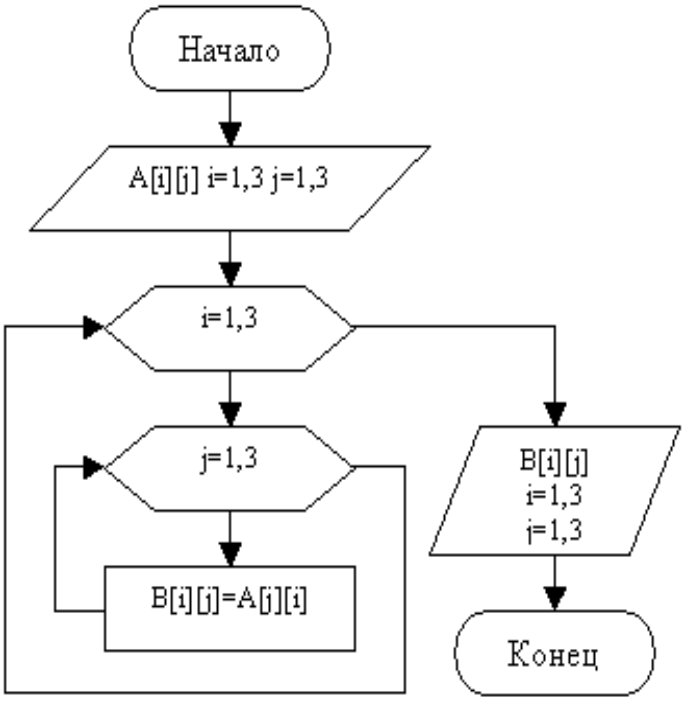
Вывод двумерного массива в виде таблицы:

```
for (i=0;i<3;i++)
```

```
{
  for (j=0;j<3;j++)
    cout<< mass [i][j]<<” “;
      cout<<endl;
}
```

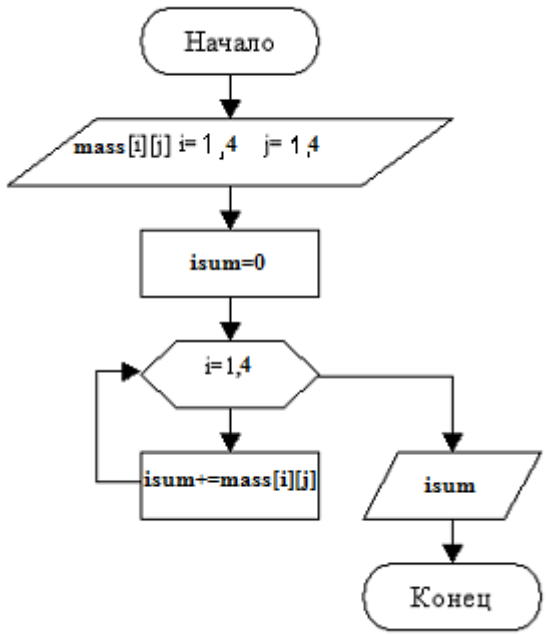
Алгоритм транспонирования двумерного массива

Дана матрица	**Получить транспонированную матрицу**
$A = \begin{pmatrix} 10 & 11 & 12 \\ 25 & 23 & 26 \\ 31 & 38 & 40 \end{pmatrix}$	$A^T = \begin{pmatrix} 10 & 25 & 31 \\ 11 & 23 & 38 \\ 12 & 26 & 40 \end{pmatrix}$




```
#include <iostream.h>
int main() {
  int A[3][3],B[3][3],i,j;
  for (i=0;i<3;i++)
    for (j=0;j<3;j++)
      cin>>A[i][j];
  for (i=0;i<3;i++)
    for (j=0;j<3;j++)
      B[i][j]=A[j][i];
  for (i=0;i<3;i++) {
    for (j=0;j<3;j++)
      cout<<A[i][j]<<” “;
    cout<<endl;
  }
  return 0;
}
```

Пример: Сумма элементов главной диагонали матрицы




```
#include <iostream.h>
void  main() {
  int mass[4][4],isum=0,i,j;
 for (i=0; i<3; i++)
     for (j=0; j<3; j++)
        cin>>mass[i][j];
             isum+=mass[i][j];
if(i==j)

cout<<"Sum="<<isum<<endl;
  }
}
```

Задания для самостоятельной работы

1. Дан двумерный массив, содержащий 3 строки и 4 столбца. Упорядочить массив по убыванию элементов 3-ей строки.

2. Дан двумерный массив, содержащий 5 строк и 2 столбца. Упорядочить массив по возрастанию элементов 2-го столбца.

3. Дана действительная квадратная матрица. Заменить нулями все элементы, расположенные на главной диагонали и выше нее.

4. Задан двумерный массив, содержащий 3 строки и 4 столбца.

Найти наибольший элемент массива, номер строки и столбца, в которых он расположен.

5. Определить количество положительных элементов каждого столбца двумерного массива, содержащего 5 строк и 5 столбцов.

6. Найти наибольший элемент главной диагонали матрицы размером 4x4 и вывести на печать всю строку, в которой он находится.

7. Найти сумму элементов массива, тип данных int, размерность 4x4.

8. Найти сумму элементов массива размерностью 4x4 расположенных выше и ниже главной диагонали.

9. Дан массив размерностью 4x4, найти сумму 2-ой и 3-ей строки.

10. Дан массив 4x4заменить все 0 на 1 в этом массиве.

Литература

1. Харви Дейтел, Пол Дейтел. Как программировать на C++. Пер. с англ. - Москва: ЗАО "Издательство БИНОМ", 1998. 1024с.
2. Марченко А.Л. C++. Бархатный путь
3. М. Эллис, Б. Страуструп. Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992. 445с.
4. Э.А.Ишкова. C++. Начала программирования – М.: ЗАО «Издательство БИНОМ»,2000. 304 с.
5. Стенли Б. Липпман. C++ для начинающих: Пер. с англ. 2тт. - Москва: Унитех; Рязань: Гэлион, 1992, 304-345сс.
6. Бруно Бабэ. Просто и ясно о Borland C++: Пер. с англ. - Москва: БИНОМ, 1994. 400с.
7. В.В. Подбельский. Язык C++: Учебное пособие. - Москва: Финансы и статистика, 1995. 560с.
8. Т. Сван. Освоение Borland C++ 4.5: Пер. с англ. - Киев: Диалектика, 1996. 544с.
9. Г. Шилдт. Самоучитель C++: Пер. с англ. - Санкт-Петербург: BHV-Санкт-Петербург, 1998. 620с.
10. У. Сэвитч. C++ в примерах: Пер. с англ. - Москва: ЭКОМ, 1997. 736с.

Корректор ***Эркинбек к. Ж.***
Редактор ***Турдукулова А.К.***
Тех.редактор ***Кочоров А.Д***

Подписано к печати 10.04.2015 г. Формат бумаги 60x84$^{1}/_{16}$.
Бумага офс. Печать офс. Объем 1,75 п.л. Тираж 50 экз. Заказ 202. Цена 42,5с.
Бишкек, ул. Сухомлинова, 20. ИЦ “Текник” КГТУ им. И.Раззакова, т.: 54-29-43
e-mail: *beknur@mail.ru*