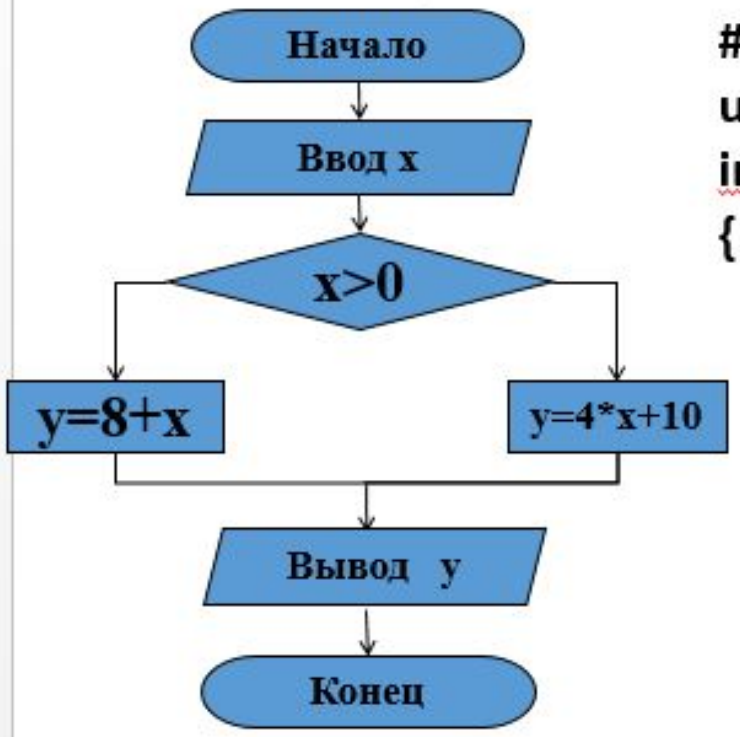
/Линейный алгоритм/

```
#include<iostream>
using namespace std;
int main ()
{setlocale(LC_CTYPE,"Russian");
float a, b, c, P;
cout<<«a=";
cin>>a;
cout<<«b=";
cin>>b;
cout<<“c=";
cin>>c;
P=(a+b+c)/3;
cout<<"P="<<P;
system("pause");
return 0;
}
```

Алгоритм с ветвлением

$$\begin{cases} 4x + 10, \text{если } x > 0; \\ 8 + x, \text{если } x < 0 \end{cases}$$




```
#include <iostream>
using namespace std;
int main()
{
   float x,y;
   cout<<"Vvedite x = "; cin>>x;
   if (x>0)
   {y=4*x+10;}
   else
   {y=8+x;}
   cout<<"Rezultat Y = "<<y<< endl;
   return 0;
}
```

В C++ существует три логические операции:

1. Логическая операция И `&&`, нам уже известная;
2. Логическая операция ИЛИ `||`;
3. Логическая операция НЕ `!` или логическое отрицание.

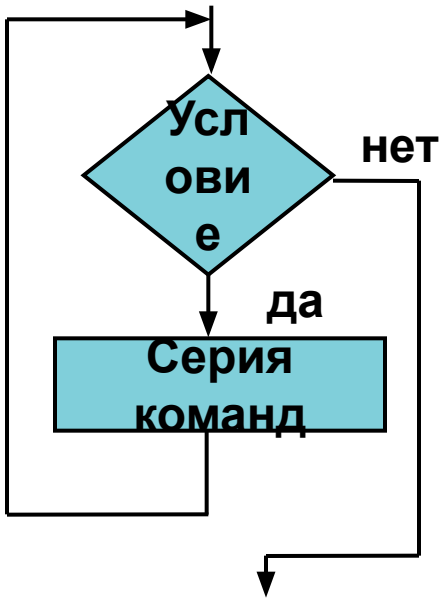
Логические операции образуют сложное (составное) условие из нескольких простых (два или более) условий. Эти операции упрощают структуру программного кода в несколько раз. Да, можно обойтись и без них, но тогда количество ифов увеличивается в несколько раз, в зависимости от условия. В следующей таблице кратко охарактеризованы все логические операции в языке программирования C++, для построения логических условий.

Таблица 1 — Логические операции C++

Операции	Обозначение	Условие	Краткое описание
И	**&&**	a == 3 **&&** b > 4	Составное условие истинно, если истинны оба простых условия
ИЛИ	**||**	a == 3 || b > 4	Составное условие истинно, если истинно, хотя бы одно из простых условий
НЕ	**!**	**!**(a == 3)	Условие истинно, если a не равно 3

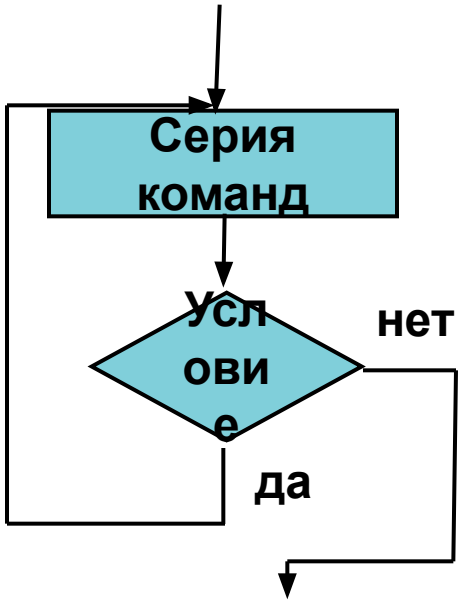
Виды циклических алгоритмов

Цикл с предусловием



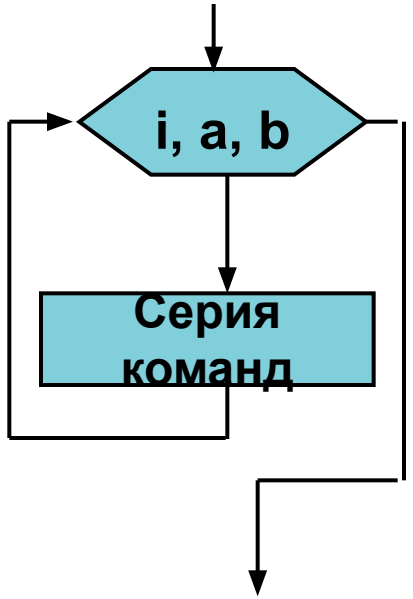
Цикл типа ***While***

Цикл с постусловием



Цикл типа
do…while

Цикл с параметром



Цикл типа ***for***

```
while (условие)
{
    Серия команд;
    }
```

Обратите внимание

1. Цикл заканчивается, когда **условие** становится **не верным (ложным).**
2. Если **условие** с самого начала ложно, то серия команд **не выполняется ни разу**.

```
do
{
Серия команд;
}
while (условие) ;
```

Обратите внимание

- Серия команд между **do** и **while** выполняется **хотя бы один раз**.
- Цикл заканчивается, когда условие становится не верным (ложным).

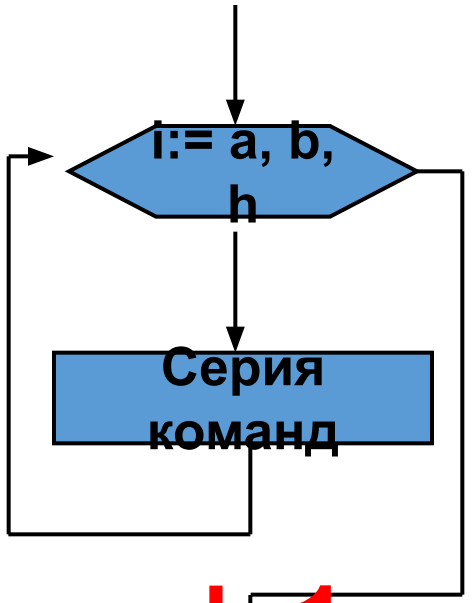
Цикл с параметром (типа «for»)

for (счетчик = значение; счетчик < значение; шаг цикла)

начало цикла

Серия команд;

конец цикла



Шаг цикла h+1

```
for  (i=0; i<n; i++)
   {
 Серия команд;
}
```

Шаг цикла h-1

```
for (i=0; i<n; i--)
   {
 Серия команд;
}
```

Трассировочная таблица

- **Трассировочная таблица** – это модель работы процессора при выполнении программы.
- Для того чтобы понять, какие значения принимают переменные на каждом шаге выполнения программы, строят трассировочные таблицы.

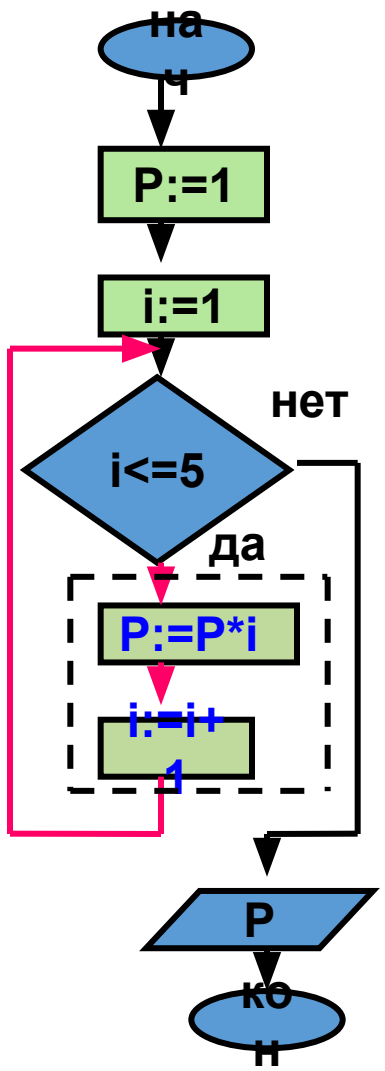
Пример:

Вычислить произведение чисел от 1 до 5 используя различные варианты цикла. Построить трассировочную таблицу.

Математическая модель:

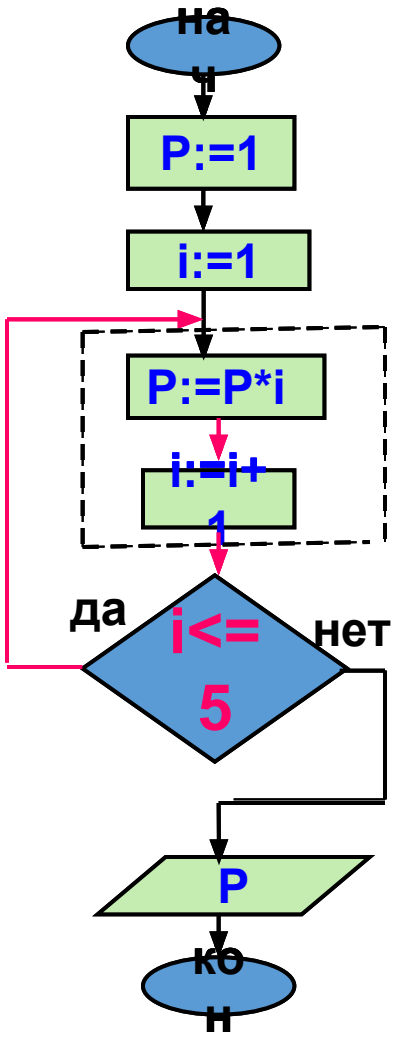
$$P = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 = 120$$

Цикл «While»



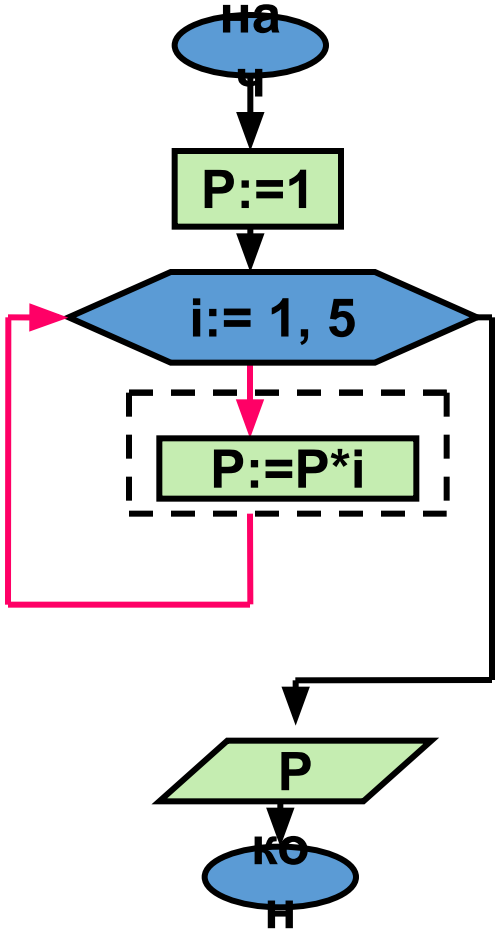
Шаг	Операция	P	i	Проверка условия
1	P:=1	1		
2	i:=1;	1	1	
3	i<=5 P:=P*i i:=i+1	1	1	1<=5, да (истина)
4	i<=5 P:=P*i i:=i+1	2	2	2<=5, да (истина)
5	i<=5 P:=P*i i:=i+1	6	3	3<=5, да (истина)
6	i<=5 P:=P*i i:=i+1	24	4	4<=5, да (истина)
7	i<=5 P:=P*i i:=i+1	120	5	5<=5, да (истина)
8	i<=5 P:=P*i i:=i+1			6<=5, нет (ложь)

Цикл «do…while»



Шаг	Операция	P	i	Проверка условия
1	P:=1;	1		
2	i:=1;	1	1	
3	P:=P*i; i:=i+1; i<5	1	1	2<5, да (истина)
4	P:=P*i i:=i+1 i<5	2	2	3<5, да (истина)
5	P:=P*i i:=i+1 i<5	6	3	4<5, да (истина)
6	P:=P*i i:=i+1 i<5	24	4	5<=5, да (истина)
7	P:=P*i i:=i+1 i<5	120	5	6<=5, нет (ложь)

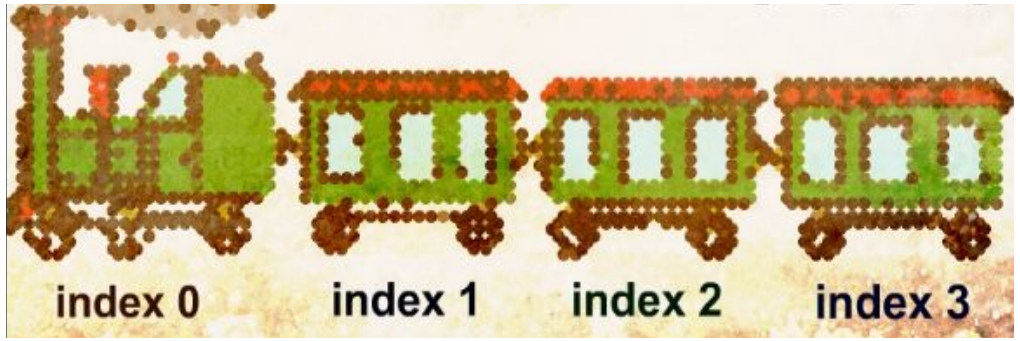
Цикл «for»



Шаг	Операция	P	i	Проверка условия
1	P:=1	1		
2	i:=1 P:=P*i	1	1	
3	i:=2 P:=P*i	2	2	
4	i:=3 P:=P*i	6	3	
5	i:=4 P:=P*i	24	4	
6	i:=5 P:=P*i	120	5	

Что такое массив?

Массив *– это линейная структура данных, в основном используется для хранения аналогичных данных. Массив представляет собой особый способ хранения элементов индексированных данных.*

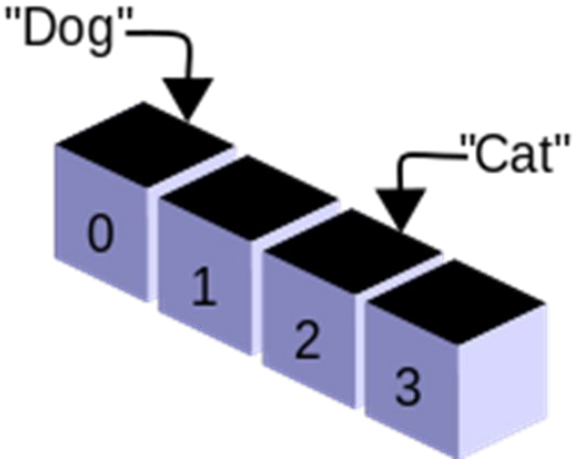


Особенности:

- ✔ *все элементы имеют один тип;*
- ✔ *весь массив имеет одно имя;*
- ✔ *элементы упорядочены по индексам (номерам).*

Что такое одномерный массив?

Одномерный массив *– это фиксированное количество элементов одного и того же типа, объединенных общих именем, где каждый элемент имеет свой номер.*



Index	0	1	2	3
Value	Nurbek	Aiym	Asyl	Erbol

Нумерация элементов массива в C++ начинается с нуля, то есть если массив состоит из 4 элементов, то его элементы будут иметь следующие номера: 0,1,2,3.

Отличие переменной от массива

Отличие переменной от массива:

Она хранит в себе только одно значение, а массив может хранить в себе несколько значений одного типа.

Структура одномерного массива

- *Элементы массива упорядочены (урегулированы) по **индексам**. Доступ к каждому элементу осуществляется путем **индексирования**, т. е. указания имени массива и номера элемента.*

Пример: a[1] = 15.

Повторение массивов
Индивидуальная работа

Дополните недостающие части кода

```
1  #include<iostream>
2  using namespace std;
3  int main()
4  {
5  int n, i;
6  cout<<"Enter the size of array:"<<endl;
7  cin>>n;
8                          //объявление массива array с типом int
9  cout<<"Write the elements of array using Enter:"<<endl;
10 for(i=0; i <    ; i++)
11                         //ввод элементов одномерного массива
12 for(i=0; i <    ; i++)
13                         //вывод элементов одномерного массива
14 return 0;
15 }
```

Правильный ответ

```
#include<iostream>
using namespace std;
int main()
{
int n, i;
cout<<"Enter the size of array:"<<endl;
cin>>n;
int array[n];          //объявление массива array с типом int
cout<<"Write the elements of array using Enter:"<<endl;
for(i=0; i < n; i++)
cin>>array[i];         //ввод элементов одномерного массива
for(i=0; i < n; i++)
cout<<array[i]<<" "; //вывод элементов одномерного массива
return 0;
}
```

Задание №1

1-группа. ***Дан одномерный массив D[5]. Выпиши имя массива и поясни, что означает цифра 5.***

a)	1
b) 5 – это …	1

2-группа. ***Объявлен одномерный массив B. Выпиши, что означает эта запись и чему равны элемент B[0]*** ***B[5] = { 7, 8, 6, 3, 10 };***

a) Описание:	1
b) Результат B[0]:	1

3-группа. ***Алмат во время работы программы заполняет одномерный массив int A[4] именами одноклассников “Anar", “Madina", "Marat", "Aibek". Программа выдает ошибку. Объясни, почему это происходит. Приведите пример значения элемента для массива int A[4].***

a) Объяснение:	1
b) Пример:	1

Критерии для оценивания:

- ✔ Объясняет основные свойства массива
- ✔ Понимает что такое одномерный массив
- ✔ Объясняет для чего применяют массив

Заполнение элементов массива в C++
1 способ

1. тип_данных имя_массива [длина_массива];

```
#include <iostream>
 using namespace std;
int main ()
{
int i, numbers[4];
cout<<“Enter the arrays elements using the Enter button:” <<endl;
for(i = 0; i < 4; i++)
cin>>numbers[i];
for(i = 0; i < 4; i++)
cout << numbers[i]<<"  ";
return 0;
}
```

```
1 int numbers[4];
```

Ввод с клавиатуры элементов одномерного массива

Заполнение элементов массива в C++
2 способ

2. тип_данных имя_массива [длина_массива] = *{список инициализации}*;

```
1 int numbers[4] = {1,2,3,4};
```

// В этом случае выполняется инициализация и присваиваются этим числам некоторые начальные значения через фигурные скобки.

```
#include <iostream>
using namespace std;
int main (){
int i, numbers[4]={1,2,3,4};
for(i = 0; i < 4; i++)
cout << numbers[i]<<" ";
return 0;}
```

Заполнение элементов массива в C++
2 способ

2. тип данных имя_массива[] = *{список инициализации}*;

```
1 int numbers[] = {1, 2, 3, 4, 5, 6};
```

// В этом случае выделяется память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации

```
#include <iostream>
using namespace std;
 int main ()
{int i, numbers[]={1,2,3,4,5,6};
for(i = 0; i < 6; i++)
cout << numbers[i]<<" ";
return 0;}
```

Заполнение элементов массива в C++
3 способ

3. Заполнение по правилу (формуле). Например, заполнить массив **numbers** квадратами натуральных чисел от 1 до 10.

```
#include <iostream>
using namespace std;
 int main ()
{
int numbers[10];
for (int i=0; i<10; i++)
numbers[i]=(i+1)*(i+1);
cout<<numbers[i]<<“  ”;
return 0;
}
```

Заполнение элементов массива в C++
4 способ

4. Заполнение массива случайными числами. Данный способ используется для быстрого заполнения массивов, которые содержат большое количество элементов (такой массив сложно ввести с клавиатуры).
При данном способе используется генератор случайных чисел.

```
#include <iostream>
#include <cstdlib>        - библиотека для функции ввода случайных чисел

rand()
using namespace std;
 int main ()
{       int Num [20];
       for (int i = 0; i < 20; i++)
       Num[i] = 1 + rand() % 9;
cout<<Num[i]<<“  ”;
      return 0;
}
```

Функция **rand()** возвращает случайное число.

В приведенном коде массив заполняется случайными числами на отрезке **от 1 до 9**

Задание №1 *(Практические задания)*

- ✔ *Напишите программу вывода элементов массива двумя способами указанными ниже.*
- ✔ *Посмотрите разницу между ними. Объясните, как будет выглядеть массив на экране в каждом случае.*

1 фрагмент
for (int i=0; i<n; i++) cout << B[i] <<" ";
2 фрагмент
for (int i=0; i<n; i++) cout<<B[i]<<endl;

Критерии для оценивания:

Использует одномерный массив в программировании