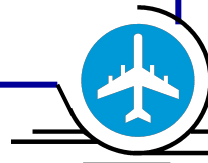


# Лекция 2.

## Ввод и вывод. ЦИКЛЫ.



# Ввод и вывод

- **Функция scanf()** вводит данные из стандартного входного потока stdin в переменные, адреса которых задаются в виде аргументов. Функция имеет переменное число параметров.

## Форматированный ввод

**scanf("<строка\_формата>", аргумент[,<аргумент>]...);**

- Пример вызова функции scanf():

```
int k; float z; char simv;  
scanf ("%c %d %f", &simv, &k, &z);
```

- Эквивалентная последовательность операторов ввода

```
scanf ("%c", &simv);  
scanf ("%d", &k);  
scanf ("%f", &z);
```



Строка описания формата ввода содержит спецификации полей формата ввода (вывода).

- Некоторые спецификации полей формата ввода (вывода):

**%d** – целое десятичное число со знаком;

**%u** – целое десятичное число без знака;

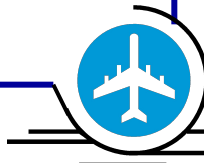
**%f** – вещественное число;

**%c** – символ;

**%s** – строка символов.

- Аргументы вызова** - адреса переменных, в которых будут храниться введенные значения.

**Символ &** - операция получения адреса переменной.



**Функция printf()** выводит символы и значения выражений в стандартный выходной поток stdout. Функция имеет переменное число параметров.

### Форматированный вывод

```
printf("<строка_формата>" [,<аргументы>]...);
```

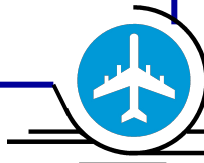
- Пример вызова функции printf():

```
printf ("\n%d%c класс - %d учеников.", t, simv, k + m);
```

Допустим, `int t=10; char simv='Б'; int k=20, m=5;`

На экране появится текст:

```
10Б класс - 25 учеников.
```



**Строка описания формата вывода** может содержать

- обычные символы,
- специальные управляющие символы,
- спецификации полей формата вывода значений переменных или выражений, если есть аргументы.

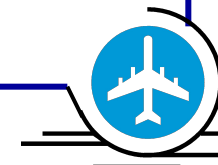
- **Некоторые специальные управляющие символы**

`\n` – символ новой строки;

`\t` – символ табуляции;

Спецификации полей формата вывода аналогичны спецификациям полей ввода.

- **Аргументы вызова** - имена переменных или выражения.



## Пример программы с вводом/выводом в стиле языка C++

// Программа 1.2. Площадь прямоугольника (в стиле C++)

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

```
void main ()
```

```
{ float a, b,           // стороны прямоугольника  
    s;                 // площадь прямоугольника
```

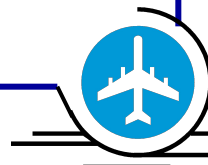
```
    cout << "\n Стороны прямоугольника = ";  
    cin >> a >> b;
```

```
    s = a * b;
```

```
    cout << "\n Площадь = " << s;
```

```
        // return ;
```

```
}
```



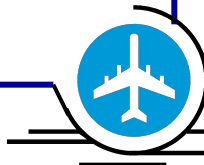
## Пояснения к программе

- **//** - **строчный комментарий**, используется в C++, наряду комментарием **/\* \*/**.
- **<iostream.h>** - **заголовочный файл**, содержит объявления стандартных объектов-потоков языка C++:
  - **cin** – поток для ввода с клавиатуры;
  - **cout** - поток для вывода на экран;
  - **>>** - операция для чтения данных из потока;
  - **<<** - операция для вывода данных в поток.



# Введение в программирование

## ЦИКЛЫ





## Циклы и ветвления

**Цикл** - для многократного повторения одних и тех же действий над данными.

- **Задача 2.1.** Составить программу табулирования функции  $f(x) = 3 + 2x$  ( шаг табулирования = 0.1).

```
/* Программа 2.1. Табулирование функции (в стиле C) */
#include <stdio.h>
main ()
{ int n, i;          /* количество строк, номер строки */
  float x;          /* текущее значение x */
  scanf ("%f %d", &x, &n);
  printf ("\n X      F(X) ");
  i=1;
  while (i <= n)
  {
    printf ("\n%5.2f  %5.2f", x, 3+2*x);
    x = x + 0.1; i = i + 1;
  }
  return 0;
}
```



## Табулирование функции

- **Тест.** Вычислить 5 значений функции, начиная с  $x = 0.2$ .

- **Вход:** 0.2 5

- **Выход:**

X	F(X)
0.20	3.40
0.30	3.60
0.40	3.80
0.50	4.00
0.60	4.20

- *Пояснения к программе.*
  1. Вводится начальное значение величины  $x$  и количество строк.
  2. Выводится заголовок таблицы значений.
  3. **while - оператор цикла** с предусловием. Позволяет повторить действия в  $\{ \}$   $n$  раз, т.е. вычислить и вывести  $n$  строк со значением функции. Переменная  $i$  – счетчик текущей строки.



Оператор цикла *while*

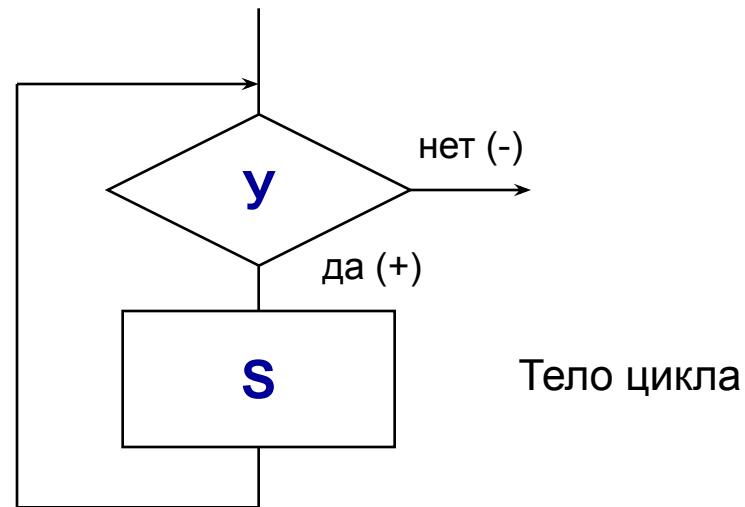
- **while** – оператор цикла с предусловием

**while** (выражение - **Условие**)

оператор **S**;

Оператор S повторяется  $\geq 0$  раз, пока значение выражения  $\neq 0$ , т.е. пока истинно условие цикла.

- **Схема работы** оператора цикла *while*



- **Условие** в циклах и ветвлениях - **выражение** языка C в скобках.

- Значение выражения должно быть целым числом.

*Условие истинно*, если *значение выражения  $\neq 0$* , *ложно* в противном случае.

- Примеры условий:

`while ( a>0 )` истинно, если  $a > 0$

`while ( a )` истинно, если  $a \neq 0$

`while ( a!=0 )` истинно, если  $a \neq 0$

`while ( !a )` истинно, если  $a = 0$

`while ( a+b )` истинно, если  $a+b \neq 0$

`while ( a && b )` истинно, если  $a \neq 0$  и  $b \neq 0$

`while ( a > 0 || b > 0 )` истинно, если  $a > 0$  или  $b > 0$

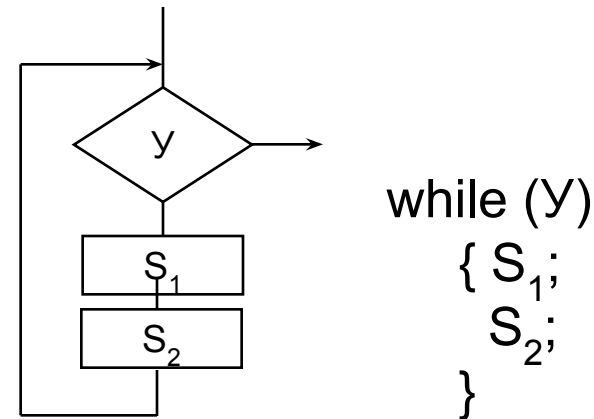
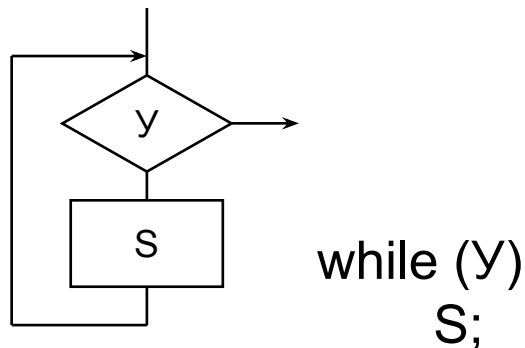


## Составной оператор

Оператор  $S$  в теле цикла может быть **простой** или **составной**.

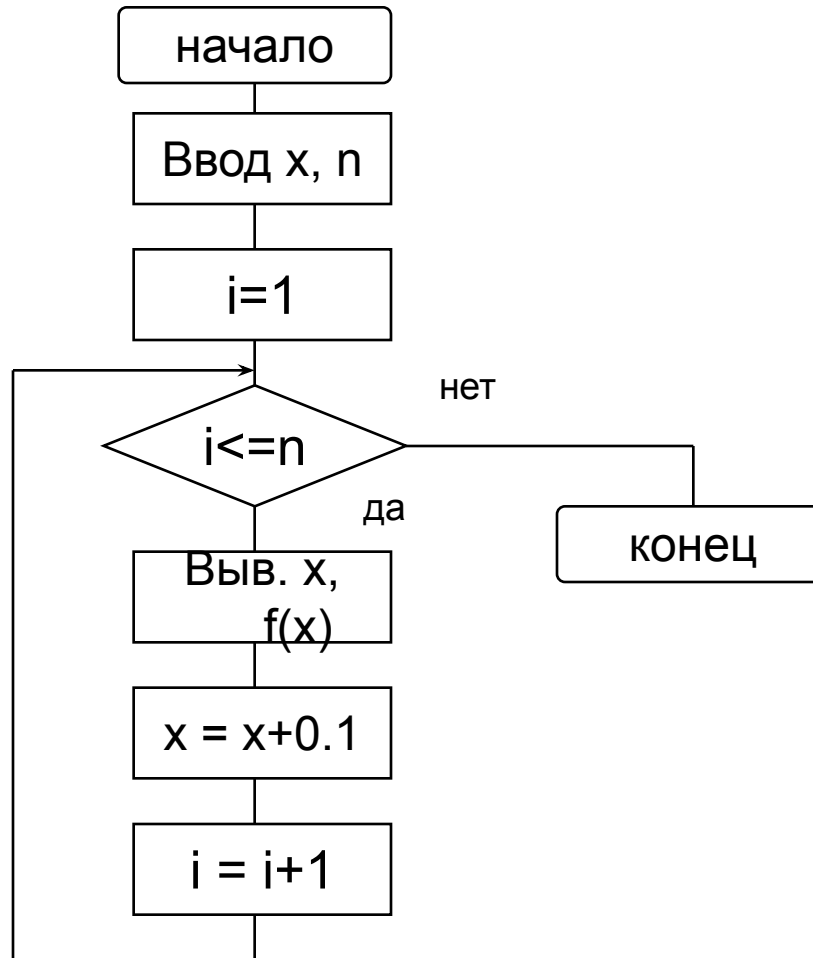
- **Составной оператор** ограничивается фигурными скобками  $\{ \}$ , состоит из одного или более операторов любого типа.

Если в теле цикла должно быть более одного простого оператора, то эти операторы нужно объединить в составной оператор.



## Схема алгоритма программы 2.1.

- Схема алгоритма программы «Табулирование функции»



## Оператор цикла do while

При  $n > 1$  возможно использование цикла с постусловием.

- **Оператор цикла с постусловием** имеет вид

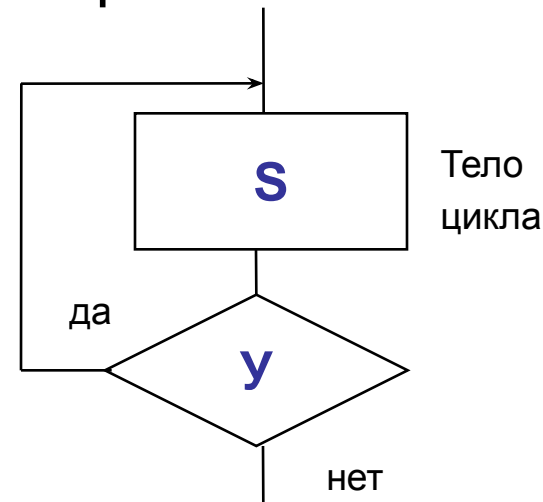
**do**

оператор **S**

**while** (выражение - Условие);

Оператор **S** повторяется  $\geq 1$  раз, пока истинно условие цикла, т.е. значение выражения  $\neq 0$ .

- **Схема работы**  
оператора цикла  
do while



## Оператор цикла for

- **Оператор цикла for** удобно использовать, если известно количество повторений цикла.

В программе «Табулирование функции» заменим оператор цикла `while` оператором `for`:

- ```
for ( i = 1; i<=n; i++)
{   printf ("\n%5.2f %5.2f", x, 3+2*x);
    x = x + 0.1;
}
```

- или короткая форма записи

```
for ( i = 1; i<=n; x += 0.1, i++)
    printf ("\n%5.2f %5.2f", x, 3+2*x);
```

, - операция последовательного вычисления.





## Оператор цикла for

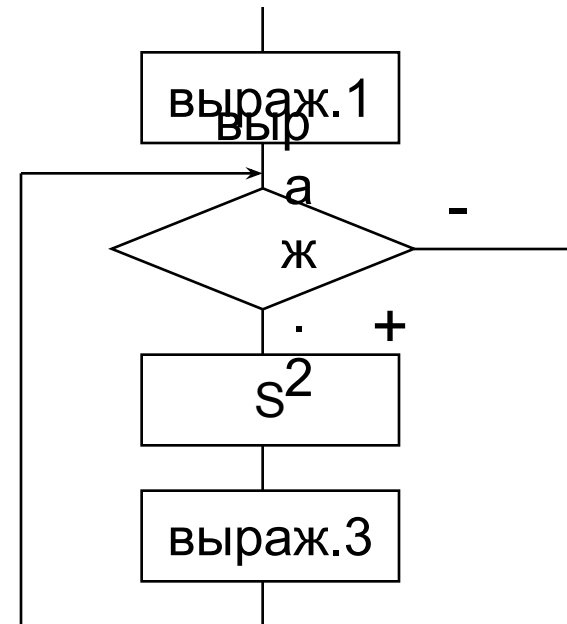
- Оператор цикла for имеет вид

```
for ([выражение1]; [выражение2]; [выражение3])  
    оператор S;
```

## Схема работы оператора цикла for

Эквивалентный  
оператор while:

```
выражение1;  
while ( выражение2 )  
{  
    оператор S;  
    выражение3;  
}
```



## Использование подпрограммы для вычисления функции $f(x)$

```
/*Программа 2.2. Табулирование функции (в стиле C) */
/* Подпрограмма вычисления функции f(x) */
#include <stdio.h>
float f ( float z )
{ float y;
  y = 3 + 2*z;
  return y;          /* возврат значения функции */
}
void main ()
{ int n, i;          /* количество строк, номер строки*/
  float x;          /* текущее значение x */
  scanf ("%f %d", &x, &n);
  printf ("\n X      F(X) ");
  i = 1;
  while (i <= n)
  {printf ("\n%5.2f  %5.2f", x, f ( x ) ); /* вызов функции */
    x = x + 0.1; i++;
  }
}
float f ( float z )
{ return 3 + 2*z; } /* возврат значения функции */
```

