

Введение в программирование

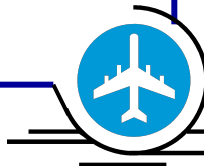
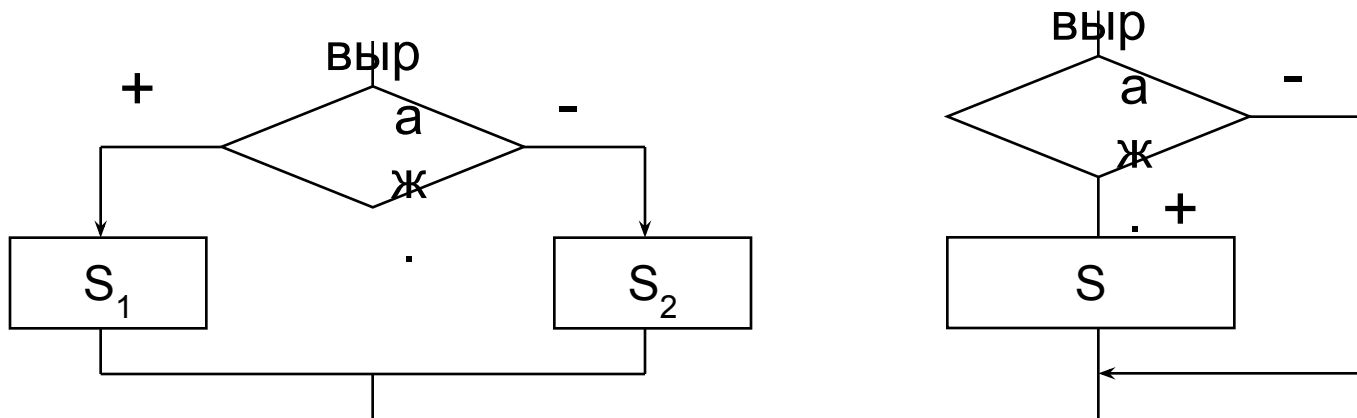
Лекция 3.

ВЕТВЛЕНИЯ. ПОСЛЕДОВАТЕЛЬНАЯ ОБРАБОТКА ДАННЫХ



Ветвления

- **Условный оператор if** для записи ветвлений:
if (выражение) оператор S_1 [**else** оператор S_2]
- **Сокращенный условный оператор:**
if (выражение) оператор S
- **Схема работы** оператора **if** (полного и сокращенного)



Ветвления

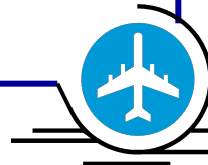
- **Условие ветвления** алгоритма и программы – **выражение языка C** в скобках (так же как в циклах). **Условие истинно**, если **значение выражения $\neq 0$** , **ложно** в противном случае. Оператор `s` может быть простым или составным.

- Требуется вычислить значение величины

$$y = \begin{cases} a*x^2 + b^2*x, & \text{если } a < 0 \\ x - a*b, & \text{если } 0 \leq a < 1 \\ 1 + x, & \text{если } a \geq 1 \end{cases}$$

/ фрагмент программы для вычисления величины y */*

```
float a, b, x;  
cin >> a >> b >> x;  
if ( a < 0 ) cout << "y = " << a *x *x + b *b *x;  
else if ( a < 1 ) cout << "y = " << x - a * b;  
else cout << "y = " << 1 + x;
```



Последовательная обработка

- **Последовательная обработка данных** (однопроходная обработка) применяется если:

1. необходимо вводить и обрабатывать **последовательность элементов** исходных данных, **в том порядке, в каком она размещена** в файле на внешнем носителе;
2. **каждый элемент** последовательности используется **не более одного раза**.

Не требуется хранения сразу всех элементов. **Достаточно иметь одну переменную**, содержащую текущий (очередной) элемент входной последовательности.

В некоторых случаях используются несколько текущих элементов (например, два-три соседних).



Последовательная обработка

- Элементами данных последовательности могут быть:
 - числа;
 - символы, строки;
 - записи файла и др.
- **Последовательность исходных данных может задаваться:**
 1. с указанием количества элементов;
 2. с признаком конца последовательности;
 3. обрабатываться до конца входного файла.



Последовательная обработка

- 1. Входная последовательность задается с указанием количества элементов $n \geq 0$ в следующем порядке:

$$n, X_1, X_2, \dots, X_n$$

- Алгоритм 3.1.
Последовательная обработка заданного количества элементов.

Ввод n ;

for ($j=1$; $j \leq n$; $j++$)

```
{  Ввод  $X$ ;  
  Обработка  $X$ ;  
}
```



Последовательная обработка

- 2. Входная последовательность задается с признаком конца:

$$X_1, X_2, \dots, X_n, W$$

где n - неизвестное заранее количество элементов ($n \geq 0$), W - признак конца последовательности (известное заранее значение, отличающееся от элементов последовательности).

- Алгоритм 3.2. Последовательная обработка элементов с признаком конца W .

Ввод X ;

while ($X \neq W$)

{ Обработка X ;

Ввод X ;

}



Последовательная обработка

- Если количество повторений $n > 0$, то возможно использование цикла с постусловием

- Ввод X ;

do

{ Обработка X ;

Ввод X ;

}

while ($X \neq W$);



Последовательная обработка

- 3. Входная последовательность

X_1, X_2, \dots, X_n продолжается до конца входного файла,

n - неизвестное заранее количество элементов ($n \geq 0$),
признак конца отсутствует.

- Алгоритм 3.3. Последовательная обработка элементов до конца файла.

Ввод X ;

while (не конец входного файла)

```
{  Обработка  $X$ ;  
  Ввод  $X$ ;  
}
```



Последовательная обработка

- **Конец файла** при вводе с клавиатуры задается комбинацией клавиш **Ctrl+Z** и **Enter**.

В программе на языке C конец входного файла можно обнаружить после попытки ввода данных за пределами файла.

Например, **значением функции scanf** является **количество фактически введенных элементов**. Если не удалось ввести ни одного элемента, значение функции равно -1.



Последовательная обработка

- Иногда в языке С можно обойтись одной операцией ввода X , поместив ее внутри условия цикла.

```
while (ввод  $X$  и  $X \neq W$ )  
    Обработка  $X$ ;
```

```
while (ввод  $X$  и не конец входного файла)  
    Обработка  $X$ ;
```



Сумма числовой последовательности

• **Задача 3.1.**• **«Сумма числовой последовательности».**

- Определить сумму и среднее арифметическое значение элементов заданной числовой последовательности.
- а) Входная последовательность задается с указанием количества элементов $n \geq 0$ в следующем порядке:

n x_1 x_2 ... x_n

- **Тест.** Вход: 4 2 3.5 1.5 5

Выход: Сумма = 12.00

Среднее = 3.00



Сумма числовой последовательности

- Используем алгоритм 3.1. для определения суммы элементов числовой последовательности.

X – текущий элемент последовательности,

S – текущее значение суммы элементов.

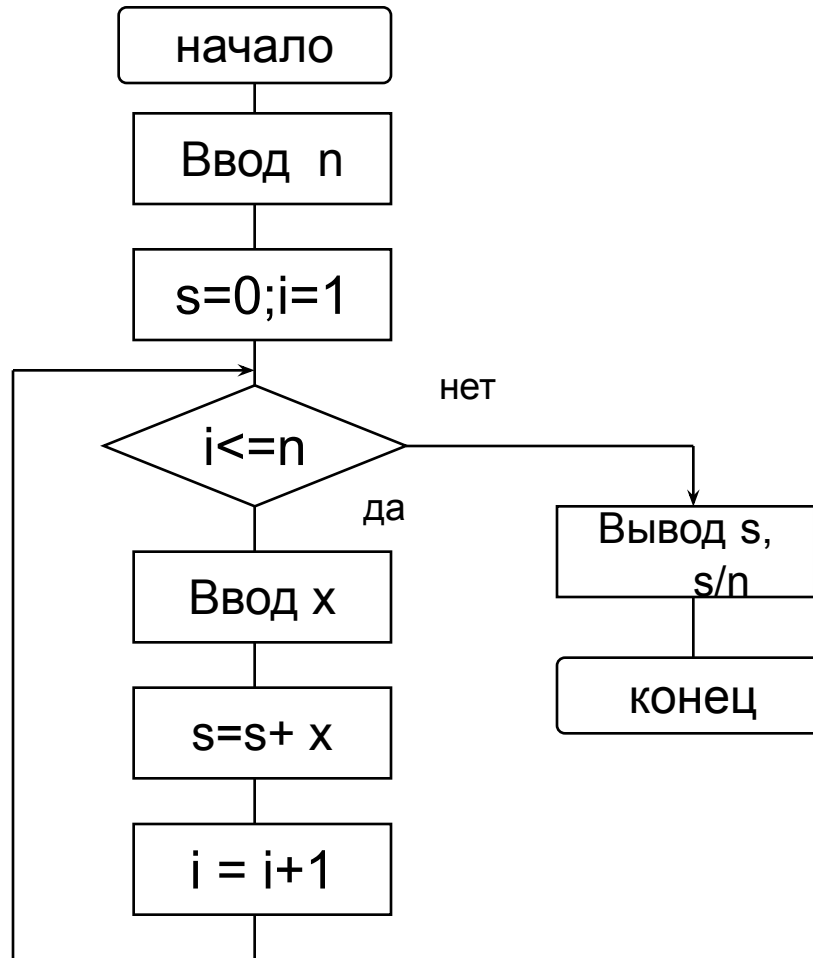
До начала цикла по вводу и обработке элементов величину s обнуляем.

На каждом шаге цикла вводим одно очередное число и его значение добавляем к величине s .



Сумма числовой последовательности

Схема алгоритма «Сумма числовой последовательности»



Сумма числовой последовательности

Трассировочная таблица исполнения алгоритма задачи
«Сумма числовой последовательности»

n=	4																	
s=	0			2.0			5.5			7.0				12.0				
i=		1			2				3				4					5
x=				2			3.5			1.5				5				
i<=n			+			+			+				+					-

Вывод

Сумма=12.00

Среднее=3.00



Сумма числовой последовательности

```
/* Программа 3.1а. Сумма и среднее числовой последовательности.*/
/*   Задано количество чисел n>=0.                               */
#include <stdio.h>
void main(void)
{ int n;                /* Количество чисел      */
  float x,              /* Текущее число      */
        sum=0;         /* Текущая сумма      */
  int i;               /* Номер текущего числа */
  scanf("%d", &n);
  for ( i = 1; i <=n; i++)
    { scanf("%f", &x);    /* Ввод текущего числа */
      sum = sum + x;
    }
  printf ("\n Сумма = %.2f\n Среднее = %.2f\n", sum, sum/n);
}
```



Сумма числовой последовательности

```
/* Программа 3.16. Сумма и среднее числовой последовательности.*/
/* Задан признак конца W = 9999 */
#include <stdio.h>
#define W 9999
void main(void)
{ float x, /* Текущее число */
  sum=0; /* Текущая сумма */
  int k = 0; /* Количество чисел */
  scanf ("%f", &x);
  while (x != W)
  { sum = sum + x; k++;
  scanf ("%f", &x);
  }
  if (k > 0)
  printf ("\n Сумма = %.2f\n Среднее = %.2f\n", sum, sum/k);
  }
```



Сумма числовой последовательности

```
/* Программа 3.1в. Сумма и среднее числовой последовательности.*/  
/* Последовательность продолжается до конца файла */  
#include <stdio.h>  
void main(void)  
{ float x, /* Текущее число */  
  sum=0; /* Текущая сумма */  
  int k = 0; /* Количество чисел */  
  while (scanf ("%f", &x) >0) /* ввод чисел до конца файла */  
  { sum = sum + x; k++;}  
  if (k>0)  
  printf ("\n Сумма = %.2f\n Среднее = %.2f\n", sum, sum/k);  
}
```



Максимум числовой последовательности

• Задача 3.2.

«Максимум числовой последовательности»

Последовательность вещественных чисел продолжается до конца файла. Составить программу нахождения максимального члена последовательности.

- **Тест.** Вход: -5 3.1 2
 Выход: Максимум = 3.100000



Максимум числовой последовательности

- Используем алгоритм 3.3. для обработки последовательности.

x – текущий элемент последовательности,

max – максимум просмотренной части последовательности.

Начальное значение **max** равно первому члену последовательности.

На каждом шаге цикла вводим одно очередное число и если очередное число оказывается больше **max**, оно заменяет максимум.



Максимум числовой последовательности

```
/* Программа 3.2. Нахождение максимального элемента */
/* числовой последовательности */

#include <stdio.h>
void main(void)
{ float x, max;          /* Текущее число, текущий максимум */
  int k;                /* Количество введенных чисел */
  k = scanf("%f", &max); /* 1-е число */
  if (k < 1) printf ("\nВходная последовательность пуста\n");
  else
  { while ((k = scanf("%f", &x)) > 0)
    { if (x > max) max = x;
      printf ("\nМаксимум= %f\n", max);
    }
  }
}
```

