

# Введение в программирование

## Лекция 6.

# СТРУКТУРНОЕ ПРОГРАММИРОВАНИЕ СВЕРХУ ВНИЗ



## Основные этапы решения задач на ЭВМ

### 1. Проектирование программы (17%).

1.1. Постановка задачи.

1.2. Выбор или разработка метода решения.

1.3. Алгоритмизация - проектирование структуры данных и алгоритма программы.

**Программа = Данные + Алгоритм**

### 2. Программирование (8%).

### 3. Отладка программы (25%).

### 4. Сопровождение программы (50%).

Борьба с ошибками - главная проблема программирования.



## Алгоритмизация

Алгоритмизация - это представление метода решения задачи в виде четкого алгоритма.

### Методы алгоритмизации:

- *структурное программирование;*
- *разработка сверху вниз.*

Структурное программирование основано на применении так называемых структурных алгоритмов, построенных из стандартных базовых структур.

*Главная идея структурного программирования – стандартизация.*



## Базовые структуры и операторы языка С для их реализации:

- Последовательность

$S_1; S_2; \dots; S_n;$

или

$\{S_1; S_2; \dots; S_n;\}$

// составной

оператор

- Ветвление

if (условие)  $S_1$ ; else  $S_2$ ; // полное

if (условие)  $S$ ; // сокращенное

- Циклы

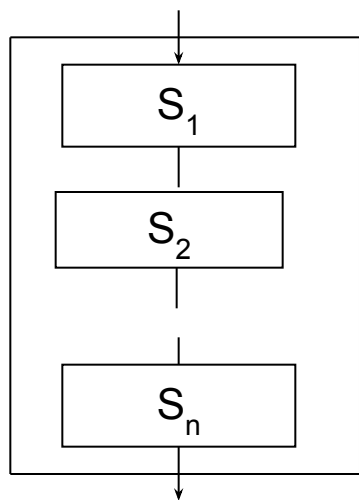
while (условие)  $S$ ; // с предусловием

do  $S$  while (условие); // с постусловием

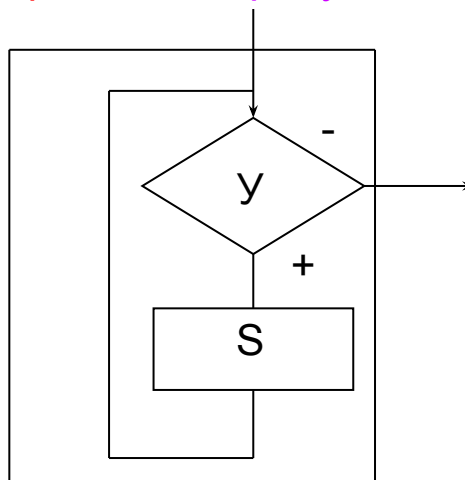


## Базовые алгоритмические структуры

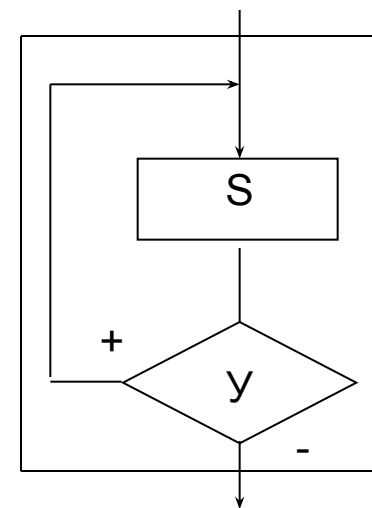
Последовательность



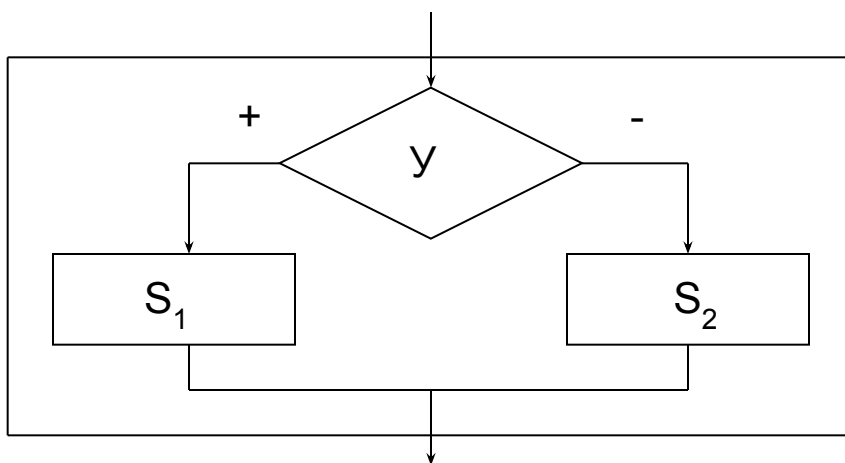
Циклы: с предусловием



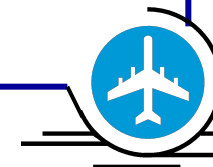
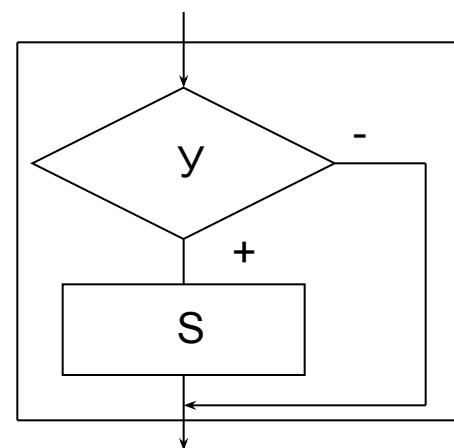
с постусловием



Ветвления: полное

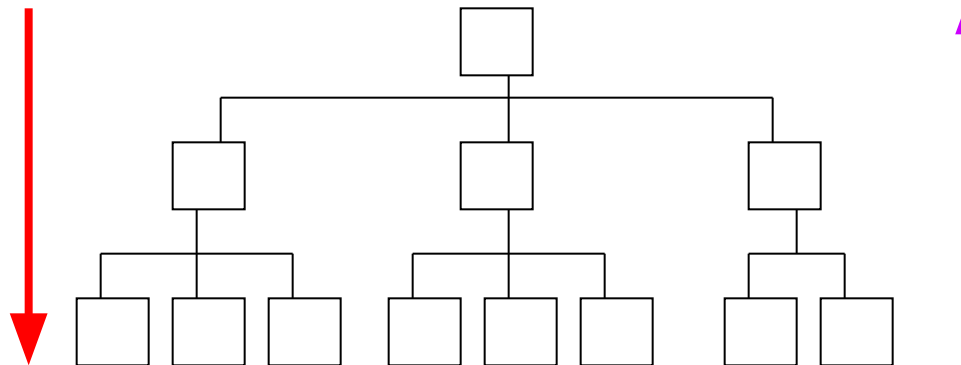


сокращенное



## Разработка сверху вниз или снизу вверх

Сложная система состоит из более мелких частей, таким же образом можно представить любой алгоритм или программу.



Сверху вниз

Снизу вверх

*Структурное программирование обычно сочетают с проектированием алгоритма сверху вниз.*



$$\cos x = x$$

**Пример 6.1.** Составить программу решения уравнения

$$\cos x = x \quad (6.1)$$

### 1. Постановка задачи.

Обозначим  $F(x) = \cos x - x$ , тогда

$$F(x) = 0 \quad (6.2)$$

Уравнение (6.1) не имеет аналитического решения, поэтому **требуется найти приближенное значение  $x$**  с погрешностью, не более заданной величины  $\epsilon$ .

Если  $F(x)$  непрерывна на отрезке  $[a; b]$  и  $F(a) \cdot F(b) < 0$ , то она имеет на этом отрезке хотя бы один корень.

Для уравнения (6.1) можно взять  **$a=0$ ,  $b=1$** .



$$\cos x = x$$

## 2. Выбор метода решения задачи

Корень можно уточнить с любой заданной погрешностью, например, **методом деления пополам**.

Находится середина исходного отрезка, и в качестве нового отрезка выбирается та его половина, где на концах функция  $F(x)$  имеет разные знаки.

Описанное разбиение отрезка пополам повторяется, пока не будет достигнута требуемая точность.

Если корней несколько, будет найден один из них.





### 3. Алгоритмизация

#### а) **УКРУПНЕННЫЙ** алгоритм

Ввод a,b,e

if (F(a)\*F(b) > 0) // Корень может отсутствовать

Вывод сообщения об ошибке

else **УТОЧНЕНИЕ** корня делением пополам



**б) УТОЧНЕНИЕ** корня делением пополам

$l=a$ ;  $p=b$ ;

$s=(l+p)/2$ ;

while ( $p-l > 2*e$ ) // Погрешность  $>e$

**РАЗБИЕНИЕ** отрезка пополам

Вывод корня  $s$ ;

**в) РАЗБИЕНИЕ** отрезка пополам

if ( $F(l)*F(s) < 0$ ) // В левой части меняется знак

$p = s$ ; // Выбор левой половины

else  $l = s$ ; // Выбор правой половины

$s=(l+p)/2$ ;



COS X = X

## Программа 6.1

```
/* Решение уравнения F(x)=0 на [a; b] */
/* с погрешностью e */
/* метод деления пополам (где F(x) = cos x - x ) */
#include <iostream.h>
#include <math.h>
/*Функция F(x) = cos x - x */
float F (float x)
{ return cos(x)-x;
}
```



COS X = X

```
void main (void)
{
    float a, b; // Концы исходного отрезка
    float e; // Допустимая погрешность
    float l, p, s; // Концы и середина текущего отрезка

    cout << "Введите границы исходного отрезка и погрешность";
    cin >> a >> b >> e;
    if (F(a) * F(b) > 0)
        cout << "\nОшибка: на концах отрезка функция одного знака";
    else // Уточнение корня делением пополам
        { l = a; p = b; s = (l + p) / 2;
          while (p - l > 2 * e) // Разбиение отрезка пополам
            { if (F(l)*F(s) <= 0) // В левой части меняется знак
              p = s; // Выбор левой половины
              else l = s; // Выбор правой половины
            }
          cout << "\nКорень = " << s;
        }
}
```



$$\cos x = x$$

- Результаты работы программы 6.1:
- Введите границы исходного отрезка и погрешность

0 1 1E-6

- Корень = 0.739085



## Задача. Сортировка числовой последовательности.

Дано целое  $n$  и вещественные  $x_1, x_2, \dots, x_n$ . Составить программу печати заданных вещественных чисел в порядке возрастания.

- **Тест:** Введите количество чисел 5

Введите числа            12 6 14 3 10

**Упорядоченные числа:** 3.0 6.0 10.0 12.0 14.0

Разработаем алгоритм нисходящим методом на псевдокоде.

Для хранения данных **нужен массив.**



- **Укрупненный алгоритм на псевдокоде имеет вид:**

**int n;** *// Количество входных чисел*

**float x[n];** *// Массив для хранения чисел*

1. Ввод массива x;
2. Сортировка массива x по возрастанию;
3. Вывод массива x;



## Сортировка чисел

Детализируем шаги алгоритма.

```
/* 1. Ввод массива x */
```

```
int i;
```

```
...
```

```
Ввод n;
```

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

```
    Ввод x[i];
```

```
/* 3. Вывод массива x */
```

```
Вывод заголовка "Упорядоченные числа:";
```

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

```
    Вывод x[i];
```





# Простейший метод сортировки – метод обмена (пузырька).



## Сортировка чисел

```
/* 2. Сортировка по возрастанию методом обмена */
```

```
...
```

```
for (k=n-1; k>0; k--)
```

```
/* Просмотр элементов x[0], ... , x[k] */
```

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

```
/* Сортировка x[i] и x[i+1] */
```

```
if (x[i] > x[i+1]) /* Порядок нарушен */
```

```
Обмен: x[i] <--> x[i+1];
```



## Сортировка чисел

```
/* Печать входных чисел по возрастанию */
#include <stdio.h>
#define NMAX 100 // Макс-е кол-во входных
чисел
void main (void)
{ float x[NMAX]; // Обрабатываемые числа

  int n; // Количество чисел

  int i; // Индекс текущего числа

  int k; // Макс. индекс просмотра
  float r; // Для обмена
```



## Сортировка чисел

```
/* 1. Ввод массива x */  
printf ("\nВведите количество чисел\n");  
scanf ("%d", &n);  
printf ("Введите числа\n");  
for (i=0; i<n; ++i)  
    scanf("%f", &x[i]);
```



## Сортировка чисел

```
/* 2. Сортировка методом обмена */
```

```
for (k=n-1; k>0; k--)  
    for (i=0; i<k; i++)  
        if (x[i] > x[i+1])  
            { r=x[i]; x[i]=x[i+1]; x[i+1]=r; }
```

```
/* 3. Вывод массива x */
```

```
printf("Упорядоченные числа:\n");  
for (i=0; i<n; ++i)  
    printf (" %4.1f", x[i]);  
}
```

