

Введение в программирование

Лекция 1.

ОСНОВНЫЕ ПОНЯТИЯ ПРОГРАММИРОВАНИЯ



ОСНОВНЫЕ ПОНЯТИЯ

Автоматизированная система =
аппаратура + **программы** + пользователи
вычислительная система

Программное обеспечение ЭВМ:
прикладное, системное и инструментальное

- **Прикладное ПО** предназначено для решения конкретных прикладных задач.
- **Системное (общее) ПО** обеспечивает функционирование вычислительной системы и необходимо для решения всех задач (основная часть СПО - **операционная система - ОС**).
- **Инструментальное ПО** – средства для разработки программ (**системы программирования**).



- **Система программирования включает:**

языки программирования, трансляторы, библиотеки программ, текстовые редакторы, редакторы связей, загрузчики, средства отладки.

Язык программирования - система обозначений для записи программ. Наиболее распространенные языки программирования Basic, Pascal, **C, C++**, Fortran, Lisp, Prolog, Ada и др. Перечислены **машинно-независимые языки высокого уровня (ЯВУ)**.

Машинная независимость языка означает возможность использовать язык для ЭВМ разных типов.

Уровень языка определяется степенью его близости к машинному языку.



Для использования языка программирования на ЭВМ необходим транслятор.

Транслятор – программа для перевода программ с одного языка на другой.

Виды трансляторов: компилятор, интерпретатор, ассемблер и др.

- **Компилятор** – анализирует и переводит программу с ЯВУ на язык, близкий к машинному, без непосредственного выполнения.

Исходный модуль - текст программы на ЯВУ.

Объектный модуль - результат компиляции.

- **Интерпретатор** – анализирует и сразу выполняет каждую команду исходной программы.



Основные понятия

- **Программирование** - наука, изучающая теорию и методы разработки, производства и эксплуатации ПО ЭВМ. До половины затрат на разработку программ требует ее отладка.
- **Отладка программы** - обнаружение ошибок в программе, их локализация и исправление.
Методы отладки – **тестирование, верификация.**
- **Тестирование** - выполнение программы вручную или на ЭВМ на контрольных примерах (тестах).
Тест - исходные данные программы вместе с ожидаемым правильным результатом работы.
Верификация - доказательство правильности программы в общем виде, по законам математики.



Алгоритм - это описание последовательности операций, направленной на решение поставленной задачи.

Основное свойство алгоритма – **дискретность**.

Операция - действие конечной продолжительности над некоторыми объектами.

Операнд - объект, участвующий в операции.

Оператор - это описание операции. Алгоритм состоит из операторов.



Способы записи алгоритма

- *Текстовый.* Универсален, наиболее распространен.
- *Табличный.* Не универсален, но удобен в отдельных случаях.
- *Графический.* Наиболее нагляден. Используется в виде схем алгоритмов и программ.

Правила оформления схем регламентируются государственным стандартом.



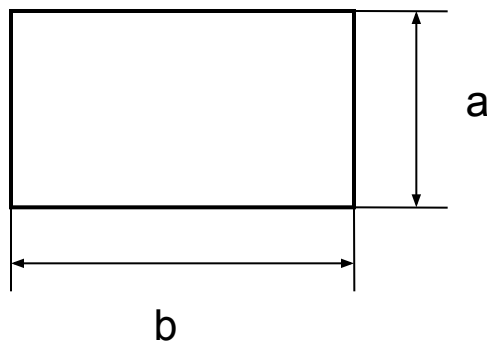
Основные символы в схемах алгоритмов:

- "**процесс**" (прямоугольник) - для описания операций ввода, обработки и вывода данных;
- "**решение**" (ромб) - для разветвления алгоритма;
- "**пуск-останов**" (овал) обозначает начало, конец и прерывание выполнения алгоритма;
- "**комментарий**" - для пояснения схемы.

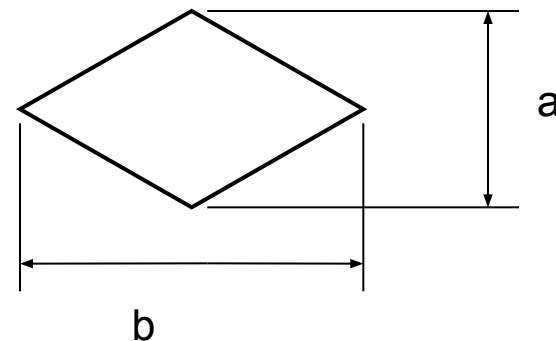


Основные символы (виды блоков) схем алгоритмов

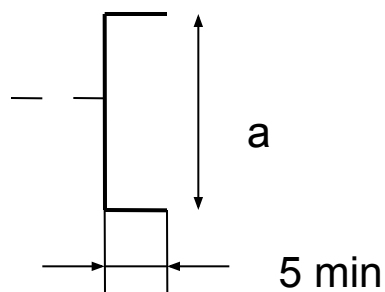
Процесс



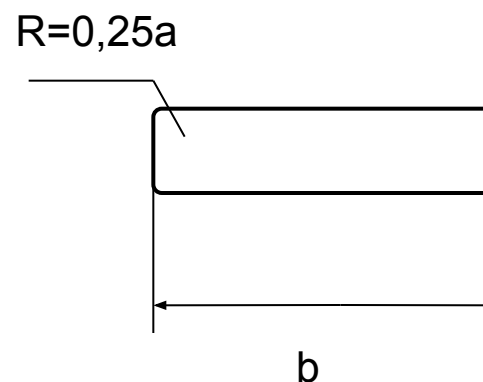
Решение



Комментарий



Пуск-останов



$a = 10, 15, 20, \dots \text{ мм}; \quad b = 1,5 a \text{ (допускается } b = 2 a \text{)}$



Пример простой программы на языке C

```
/* Программа 1.1. Площадь прямоугольника (в стиле C) */  
#include <stdio.h>  
main ()  
{ float a, b,          /* стороны прямоугольника */  
  s;          /* площадь прямоугольника */  
  printf ("\n Стороны прямоугольника = ");  
  scanf ("%f %f", &a, &b);  
  s = a * b;  
  printf ("\n Площадь = %.2f \n ", s);  
  return 0;  
}
```

Можно обойтись без дополнительной переменной s, тогда вычисление площади выполняется при выводе результата:

```
printf ("\n Площадь равна =%.2f \n ", a * b);
```



Пояснения к программе

/* */ - комментарий языка С не выполняется и не влияет на работу программы.

main() – заголовок главной функции.

Тело функции в фигурных скобках { }.

Программа на языке С состоит из одной или нескольких функций, выполнение начинается с функции main().

return - оператор возврата, завершает выполнение функции.

return 0; означает успешное завершение программы.

float a, b, s; – объявлены переменные вещественного типа.



printf () - вызов стандартной функции для вывода сообщения и результата.

```
printf ("\n Стороны прямоугольника = ");
```

Форматная строка содержит **текст в кавычках**, где символ **\n** - переход на новую строку.

```
printf ("\n Площадь = %.2f \n ", s);
```

Форматная строка, кроме текста, содержит **формат вывода переменной s**. Выводится значение переменной s или выражения $a*b$.

%f – формат вещественного числа,

%.2f – формат вывода с точностью до 2-х знаков.



Пояснения к программе

- **scanf()** - вызов стандартной функции для ввода исходных данных.

scanf ("%f %f", &a, &b);

Форматная строка содержит **форматы ввода** вещественных **переменных a, b**.

Символ & (амперсанд) перед именем переменной - **операция получения адреса переменной**.

#include <stdio.h> - директива препроцессора для использования стандартных функции ввода/вывода.

<stdio.h> - заголовочный файл, содержит объявления стандартных функций ввода/вывода.



Постоянная и переменная величина

- **Величина** имеет **обозначение**,
обладает **значением**,
принадлежит некоторому **типу данных**,
хранится по некоторому **адресу** в ОП.
- **Константа** - для представления
постоянных величин.
- Значение константы **не может измениться**.



Постоянные и переменные величины

• *Типы констант* *Примеры констант*

Целые числа: 15 -1 100 9

Вещественные числа: -1.05 0.0001 1e-4

Символьные: 'A' 'a' '*' '9'

Строковые: "KGTU" "Kazan"

• *Символические константы*

Константа может быть именована, по традиции имена констант задают заглавными буквами.

Примеры объявления символических констант:

```
#define     N     20
```

```
const float     PI = 3.1415;
```



Постоянные и переменные величины

- **Переменная величина** может принимать в программе различные значения.

До использования переменная величина должна быть объявлена.

- **Объявление переменной величины:**

<тип> <имя> [, <имя>] ...;

При объявлении определяются:

имя переменной величины – идентификатор;

тип данных, к которому она относится.

Идентификатор – имя для обозначения **переменной величины** в программе. Именуются также функции, метки.



Постоянные и переменные величины

- **Идентификатор** – последовательность латинских букв, цифр, символов подчеркивания.

Идентификатор должен начинаться с буквы или символа подчеркивания. Не допускается совпадение имени с ключевыми словами языка, например, с ключевым словом `while`.

- Примеры идентификаторов:
`summa, X, k2, k_sim`



Постоянные и переменные величины

Тип данных величины определяет:

- множество допустимых значений величины;
- набор допустимых операций над ними;
- способ представления этих значений в памяти.

Встроенные типы данных:

- Целые – **int**, **unsigned** (беззнаковое целое),
short (короткое целое), **long** (длинное целое),
char (символ);
- Вещественные – **float**, **double** (двойной точности);
- Прочие: пустой – **void**;
перечислимый – **enum**.



Целые величины

- | Тип | Размер | Диапазон значений |
|---|-----------------|---|
| • <i>unsigned char</i> | 1 байт 8 бит | от 0 до 255; |
| • <i>int</i> | 2 байта 16 бит | от -2^{15} до $2^{15}-1$,
т. е. от -32768 до 32767 |
| • <i>short</i> ($\leq int$) | 2 байта 16 бит | от -32768 до 32767 |
| • <i>unsigned</i> | 2 байта 16 бит | от 0 до $2^{16}-1$
т. е. от 0 до 65535 |
| • <i>unsigned short</i> | 2 байта 16 бит | от 0 до 65535 |
| • <i>long</i> ($\geq int$) | 4 байта 32 бита | от -2^{31} до $2^{31}-1$
т. е. приблизительно $\pm 2 \cdot 10^9$ |
| • <i>unsigned long</i> | 4 байта 32 бита | от 0 до $2^{32}-1$
т. е. приблизительно от 0 до $4 \cdot 10^9$ |



Вещественные величины

- | Тип | Размер | Точность | Диапазон значений |
|----------------------|---------|-----------------|--|
| • float | 4 байта | 6..7 знач. цифр | \pm (от 10^{-38} до 10^{+38}) |
| • double | 8 байт | 15 знач. цифр | \pm (от 10^{-308} до 10^{+308}) |
| • long double | 10 байт | 19 знач. цифр | \pm (от $3.4 \cdot 10^{-4932}$ до $1.1 \cdot 10^{+4932}$) |



Примеры объявления переменных величин

- `int i, j;`
`float x, y=0; char c;`
- `int z[5]={1, 3, 12,-8, 15};`
`float a[10][10];`
`char st[80];`
`char s[]="KGTU";`



Присваивание

Присваивание – *изменение значения* переменной величины.

Оператор присваивания имеет вид

<имя_величины> = <выражение>;

Присваивание можно понимать как операцию **«заменить на»**. Вычисляется значение выражения, которое заменяет прежнее значение переменной.

- Например, **$x = 20;$ $y = x * 10 + 5;$ $x++;$**

$x = x * x;$ $x += 10;$ $y--;$



Выражения языка СИ

- **Выражение** - это формула, определяющая последовательность операций для получения значения.
- **Выражения языка СИ** подобны алгебраическим выражениям, могут содержать константы, имена констант и переменных, вызовы функций, знаки операций, скобки.
- В языке С выражение, заканчивающееся точкой с запятой, образует **оператор-выражение**. Частными случаями этого оператора являются оператор присваивания и вызов подпрограммы.



• Например,

$$y = (1 + x * x) / (2 + x);$$
$$z = ((x + y) * 10 - 1) / 2;$$
$$i++; \quad i = i + 1;$$
$$x += 10; \quad x = x + 10;$$
$$z = \text{sqrt}(x + y) * 10;$$
$$y = (1 + \sin(x) * x) / 2;$$


•Некоторые операции языка С

Арифметические операции:

++ -- * / % + -

Операции отношений:

> >= < <= ==(равно) !=(не равно)

Логические операции:

! (не) && (и) || (или)

Операции присваивания: **= *= /= += -=**



Операции целочисленного деления

/ - частное ,

% - остаток от деления.

При *целочисленном делении* с *остатком* дробная часть частного отбрасывается.

Делимое = Частное * Делитель + Остаток

Например,

$$17/5 = 3, \text{ а } 17\%5 = 2 \quad 17 = 3*5 + 2$$



Условная операция

выражение1 ? выражение2 : выражение3

Если значение *выражения1* $\neq 0$, то результат равен *выражению2*, в противном случае - *выражению3*.

Выражение1 должно иметь целочисленное значение.

Например, нахождение **максимума** из **двух значений**:

$z = (a > b) ? a : b;$ $/* z = \max(a, b); */$



Использование стандартных функций языка C/C++

- $\sin(x)$ и \sqrt{x} - **стандартные функции языка C**:
 $\sin(x)$ – для вычисления $\sin x$;
 \sqrt{x} – для вычисления квадратного корня из x .

Для использования стандартных функций языка C (их свыше 200) необходимо **включать заголовочные файлы**.

- Например, для использования перечисленных функций **нужна директива препроцессора**

`#include <math.h>`

Заголовочный файл **`<math.h>`**

содержит **объявления математических функций**.



Использование стандартных функций языка C/C++

Заголовки некоторых стандартных математических функций:

`int abs (int i)` $|i|$ но: `abs(-32768) = -32768`

`double fabs (double x)` $|x|$

`double sqrt (double x)` корень квадратный из x

`double exp (double x)` e^x

`double log (double x)` $\ln x$

- Примеры вызова этих функций:

`int n; float x, y, z, t;`

- `n = abs (n); printf (“\n %f”, fabs(z*2));`

- `t = sqrt(y+z); x = exp(z); printf (“\n %f”, log(z/2));`

