

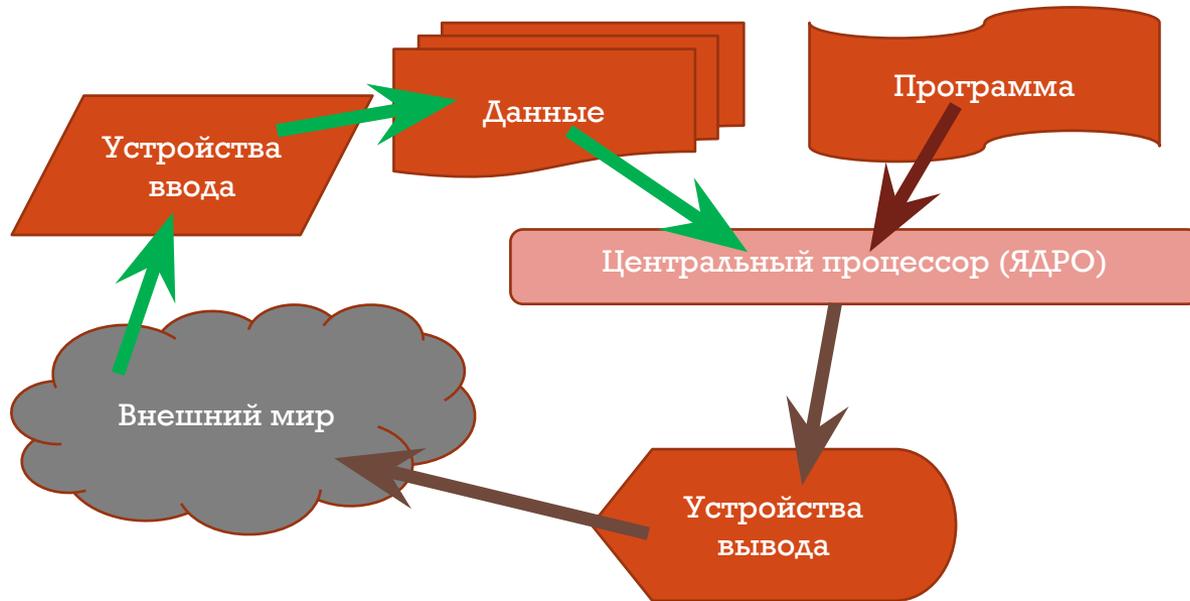
МИКРОПРОЦЕССОРН АЯ ТЕХНИКА

Практические занятия

Занятие 1



Электронно-вычислительная машина (компьютер):



Для обработки информации при помощи ЭВМ, необходимо:

- Превратить входную информацию в данные, пригодные для хранения в памяти (то есть, превратить в **числа**);
- **Расположить** данные в памяти определенным образом;
- Выполнить определенные **действия над числами**, хранящимися в памяти;
- Куда-то выдать **результат**.

Таким образом:



Язык программирования Си

- **Си** - стандартизированный процедурный язык программирования, разработанный в начале 1970-х годов сотрудниками Bell Labs Кеном Томпсоном и Денисом Ритчи
- Си – самый распространенный язык программирования для микроконтроллеров
 - высокая скорость работы программ
 - много возможностей, много готовых библиотек функций
-  стал основой многих современных языков (*C++*, *C#*, *Javascript*, *Java*, *ActionScript*, *PHP*)
 - высокие шансы сделать ошибку, которая не обнаруживается автоматически



Простейшая программа

главная (основная) функция
всегда имеет имя *main*

```
main()
```

начало
программы

«тело»
программы
(основная
часть)

```
{
```

```
}
```

конец
программы



Что делает эта программа?

Что происходит дальше?

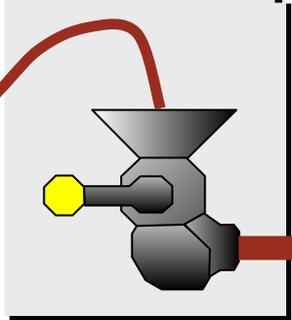
текст программы на Си

first.cpp

```
main()  
{  
  
}
```

ИСХОДНЫЙ файл

**Транслятор
(компилятор)**



first.o

```
ЪБzЦ2?|ё3БKa  
n/36ШпIC+И-  
Ц3_5MyPЧ6  
s6bd^!/:@:лЖ1_
```

объектный файл

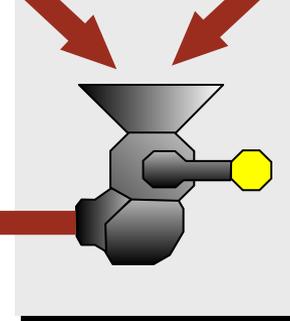
Стандартные
(готовые)
функции

first.exe

```
MZPo:€Pэ_e3"!_  
`кп,Цb€-Щр1  
G_БАС,  
_Ощяхα9жФ
```

исполняемый файл

**редактор
связей
(компоновка)**



- **исполняемый файл можно запустить**

Вывод текста на экран

include = ВКЛЮЧИТЬ

```
#include <stdio.h>
main()
{
    printf("Привет!");
}
```

файл *stdio.h*:
описание
стандартных
функций ввода
и вывода

ВЫЗОВ СТАНДАРТНОЙ
функции
printf = *print format*
(форматный вывод)

ЭТОТ ТЕКСТ
будет на
экране

Переход на новую строку

```
#include <stdio.h>
main()
{
    printf("Привет, \n Вася!");
}
```

последовательность
`\n` (код 10)
переход на новую строку

на экране:

```
Привет,  
Вася!
```

Символы управления ВЫВОДОМ

- Для управления расположением текста на экране (в окне вывода) используются специальные символы форматирования. Такие символы в тексте программы обозначаются как пара, состоящая из символа обратной косой черты «\» и строчной латинской буквы.
- Наиболее часто используются:
 - `\n` - переход на новую строку;
 - `\r` - переход в начало текущей строки;
 - `\t` - горизонтальная табуляция: прыжок вправо.

Включение русской кодовой страницы

```
#include <stdio.h>
#include <Windows.h>
main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    printf("Привет!");
}
```

файл *Windows.h*:
описание
специфических
функций
управления ОС
Windows

Вызов функций установки
номера кодовой страницы окна
консоли («черного экрана»)
1251 – кодовая страница с
символами кириллицы

Средства создания

программ

Для написания, компиляции и отладки программ в настоящее время широко используются

программные комплексы, которые называются ***IDE - Integrated Development Environment*** (интегрированная среда разработки)

- Если программа пишется и компилируется на одной платформе (например, на РС под управлением ОС Windows), а исполняется на другой платформе (например, на микроконтроллере ARM), то среда разработки называется *кроссплатформенной*

Интегрированная среда разработки

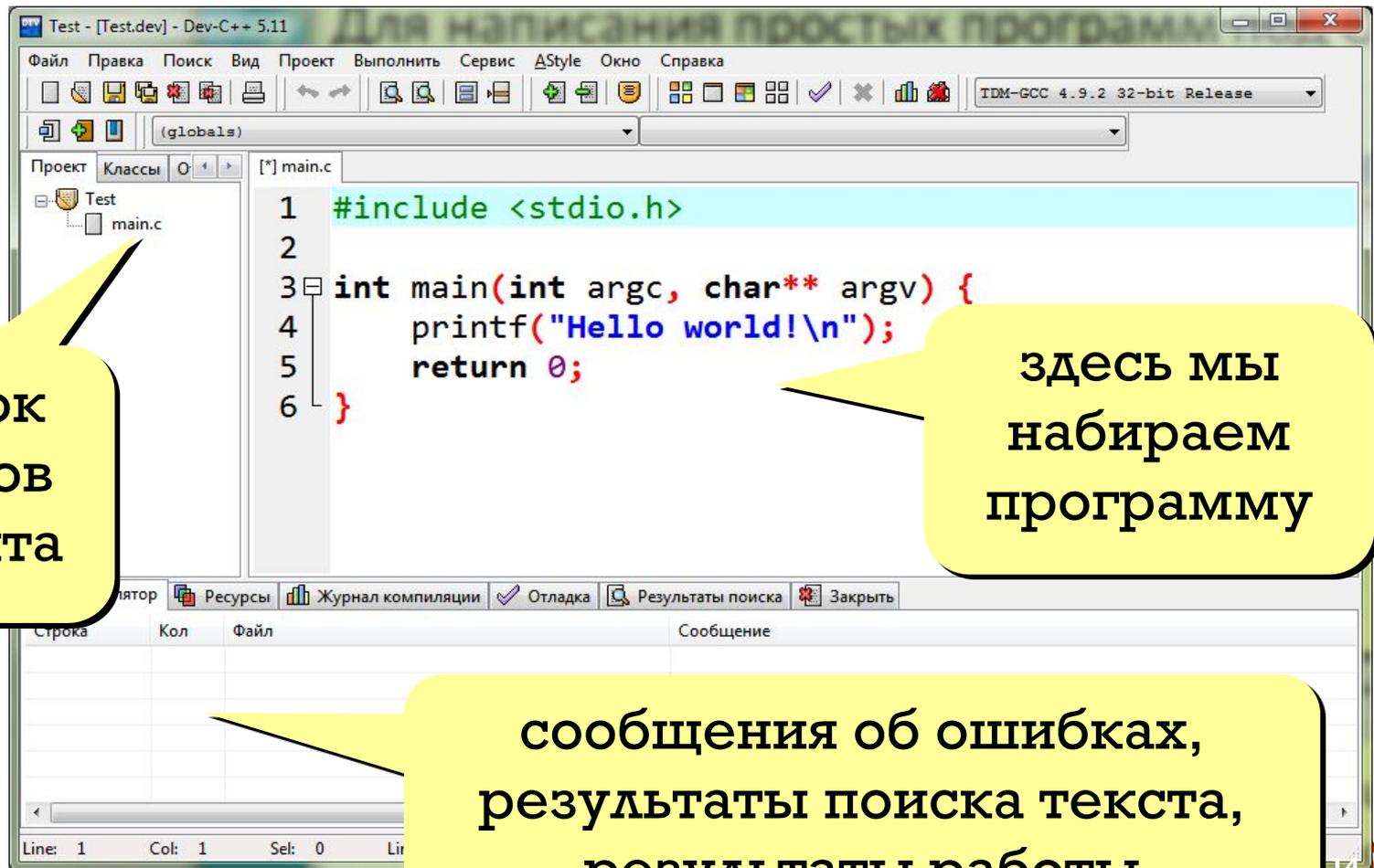
Включает в себя:

- **текстовый редактор** для создания и редактирования текстов программ
- **компилятор** для перевода текстов программ в команды процессора
- **компоновщик** для создания исполняемого файла (EXE-файла или двоичного файла, пригодного для загрузки в память микроконтроллера)
- **программатор** для записи файлов во Flash-память
- **отладчик** для поиска ошибок в программах

Интегрированная среда разработки



Для написания простых программ под ОС Windows будет использоваться IDE **DEV-CPP**



СПИСОК
файлов
проекта

ЗДЕСЬ МЫ
набираем
программу

сообщения об ошибках,
результаты поиска текста,
результаты работы

ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ

Клавиши и кнопки управления Dev-CPP

| | | |
|---|---------------------|---|
| Сохранить файл(ы) | Ctrl+S |  |
| Запуск программы (Скомпилировать и выполнить) | F11 |  |
| Отменить | Ctrl-Z |  |
| Восстановить отмененное | Shift-Ctrl-Z |  |
| Добавить отступ | Tab | |
| Убрать отступ | Shift-Tab | |

СООБЩЕНИЯ ОБ ОШИБКАХ КОМПИЛЯЦИИ

эта ошибка
обнаружена
здесь!

```
1 #include <stdio.h>
2
3 int main(int argc, char** argv) {
4     printf("Hello world!\n")
5     return 0;
6 }
```

| Ст... | Кол | Файл | Сообщение |
|-------|-----|--|--------------------------------------|
| | | C:\Projects\Stud\MPT\Test\main.c | In function 'main': |
| 5 | 2 | C:\Projects\Stud\MPT\Test\main.c | [Error] expected ';' before 'return' |
| 29 | | C:\Projects\Stud\MPT\Test\Makefile.win | recipe for target 'main.o' failed |

Line: 5 Col: 5 Sel: 0 Lines: 6 Length: 99 Вставка Done parsing in 0 seconds



Ошибка может быть в конце предыдущей строки!

НАИБОЛЕЕ «ПОПУЛЯРНЫЕ» ОШИБКИ

| | |
|---|---|
| xxx.h: No such file or directory | не найден заголовочный файл 'xxx.h' (неверно указано его имя, он удален или т.п.) |
| 'xxx' undeclared (first use this function) | имя функции или переменной 'xxx' не определено |
| missing terminating " character | не закрыты кавычки " |
| expected ; | нет точки с запятой в конце оператора в предыдущей строке |
| expected } | не закрыта фигурная скобка |

Задания

Вывести на экран текст "лесенкой"

Вася

пошел

гулять

Вывести на экран рисунок из букв

Ж

ЖЖЖ

ЖЖЖЖЖ

ЖЖЖЖЖЖЖ

НН НН

ZZZZZ

МИКРОПРОЦЕССОРН АЯ ТЕХНИКА

Практические занятия

Занятие 2



Хранение данных: Перемещение



Что такое переменная?

Переменная – это ячейка в памяти компьютера, которая имеет имя и хранит некоторое значение.

- Значение переменной может меняться во время выполнения программы.
- При записи в ячейку нового значения старое стирается.

Типы переменных

- **int** – целое число (В ОС Windows 32 - 4 байта)
- **float** – вещественное число, *floating point* (4 байта)
- **char** – символ, *character* (1 байт)
- ... и много других разных типов...

Имена переменных:

Могут включать

- латинские буквы (A-Z, a-z)
- знак подчеркивания _
- цифры 0-9

НЕ могут включать

- русские буквы
- пробелы
- скобки, знаки +, =, !, ? и др.

Какие имена правильные?

i j R&B temperature1 4you Internal_Pressure Вася
current_offset _ABBA [Privet] A+B



Объявление переменных

Объявить переменную = определить её имя, тип и начальное значение (если нужно).

```
main ()
```

```
{  
i  
f
```

целая и дробная
части отделяются
точкой

целая переменная a

вещественные
перемен

целые переменные
Tu104, Il86 и Yak42

```
int Tu104, Il86=23, Yak42;
```

вещественные
переменные x, y и z
x = 4,56

```
float x=4.56, y, z;
```

```
char c, c2='A', m;
```

символьные
переменные c, c2 и m
c2 = 'A'

```
}
```

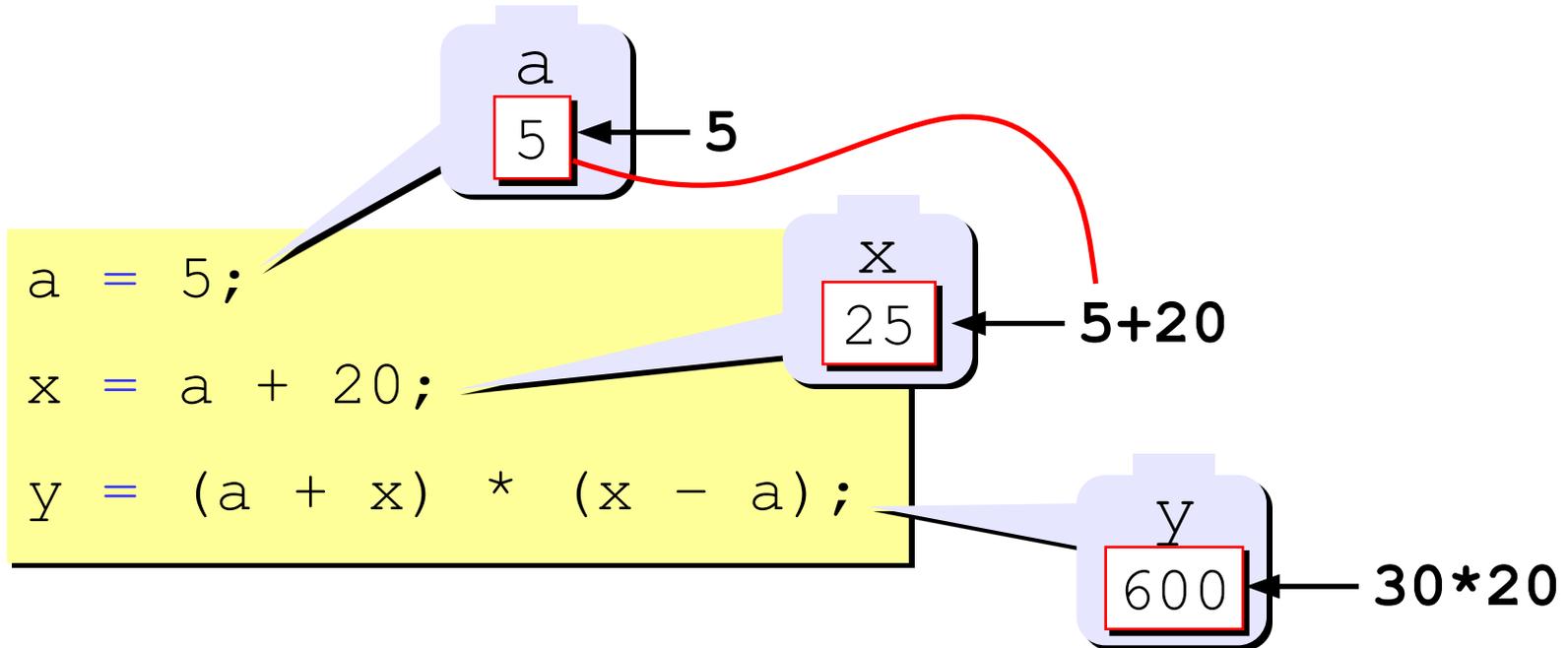


Если начальное значение не задано, то в этой переменной находится «мусор»!

Оператор присваивания

Оператор – это команда языка программирования.

Оператор присваивания служит для изменения значения переменной.



Оператор присваивания

Общая структура:

куда записать

что записать

имя_переменно = выражение;

Значение выражения вычисляется в процессе выполнения оператора присваивания.

Выражение может включать:

- константы (постоянные)
- имена переменных
- знаки арифметических операций:

+ - * / %

умножение

деление

остаток от
деления

- вызовы функций
- круглые скобки ()



Для чего служат
круглые скобки?

Какие операторы неправильные?

```
main ()
{
  int a, b;
  float x, y;
  a = 5;
  10 = x;
  y = 7, 8;
  b = 2.5;
  x = 2 * (a + y);
  a = b + x;
}
```

имя переменной
должно быть слева
от знака =

целая и дробная часть
отделяются точкой

при записи вещественного
значения в целую
переменную дробная часть
будет отброшена



Особенность деления в Си



При делении целых чисел дробная часть отбрасывается!

```
main()
```

```
{
```

```
int a = 7;
```

```
float x;
```

```
x = a / 4;
```

```
x = 4 / a;
```

```
x = (float)a / 4;
```

```
x = 4.0 / a;
```

```
}
```

=1

=0

=1.75

=0.57

Сокращенная запись операций в Си

Смысл сокращенной записи – уменьшение вероятности случайных опечаток при наборе текста программы.

| полная запись | сокращенная запись |
|--|----------------------|
| <code>a = a + 1;</code> инкремент | <code>a++;</code> |
| <code>a = a + b;</code> | <code>a += b;</code> |
| <code>a = a - 1;</code> декремент | <code>a--;</code> |
| <code>a = a - b;</code> | <code>a -= b;</code> |
| <code>a = a * b;</code> | <code>a *= b;</code> |
| <code>a = a / b;</code> | <code>a /= b;</code> |
| <code>a = a % b;</code> | <code>a %= b;</code> |

Пошаговое исполнение программы

Отладка (F5)

```
main()  
{  
    int a, b;  
    a = 5;  
    b = a + 2;  
    a = (a + 2) * (b - 3);  
    b = a / 5;  
    a = a % b;  
    a++;  
    b = (a + 14) % 7;  
}
```

| a | b |
|----|---|
| ? | ? |
| 5 | |
| | 7 |
| 28 | |
| | 5 |
| 3 | |
| 4 | |
| | 4 |

Порядок выполнения операций

- вычисление выражений в скобках
- умножение, деление, % слева направо
- сложение и вычитание слева направо

2 3 5 4 1 7 8 6 9

$$z = (5*a*c + 3*(c-d)) / a * (b-c) / b;$$

$$z = \frac{5ac + 3(c-d)}{ab} (b-c)$$
$$x = \frac{a^2 + 5c^2 - d(a+b)}{(c+d)(d-2a)}$$

2 6 3 4 7 5 1 12 8 11 10 9

$$x = (a*a + 5*c*c - d*(a+b)) / ((c+d) * (d-2*a));$$

Вычислительная программа

Задача. Ввести два целых числа и вывести на экран их сумму.

```
#include <stdio.h>
#include <Windows.h>
main()
{
    int a, b, c;
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    printf("Введите два целых
числа:\n");
    scanf ("%d%d", &a, &b);
    c = a + b;
    printf("%d", c);
}
```

подсказка для
ввода чисел

ВВОД ДВУХ
чисел с
клавиатуры

ВЫВОД РЕЗУЛЬТАТА

Ввод чисел с клавиатуры

scanf() – функция ввода с консоли

формат ввода

адреса ячеек, куда записать введенные числа

```
scanf ("%d%d", &a, &b);
```

Формат – символная строка, которая указывает, какие числа вводятся:

%d – целое число

%f – вещественное число

%c – 1 символ

%s – символная строка

ждать ввода с клавиатуры **двух** целых чисел (через пробел или *Enter*), первое из них записать в переменную *a*, второе – в *b*

&a – адрес переменной *a*

| | |
|------|----|
| ... | |
| 7653 | |
| 7652 | 12 |
| 7651 | |

12 – значение переменной *a*

Что неправильно?

```
int a, b;
```

```
scanf ("%d", a);
```

```
scanf ("%d", &a, &b);
```

```
scanf ("%d%d", &a);
```

убрать пробел

```
scanf ("%d %d", &a, &b);
```

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

&a

%d%d

&a, &b

%d%d

Вывод чисел на экран

здесь вывести
целое число

это число взять
из ячейки C

```
printf ("%d", c);
```

```
printf ("Результат: %d", c);
```

```
printf ("%d+%d=%d", a, b, c);
```

формат вывода

список значений

```
printf ("%d+%d=%d", a, b, a+b);
```

арифметическое
выражение

Вывод целых чисел

```
int x = 1234;  
printf ("%d", x);
```

или "%i"

1234

минимальное число
позиций (+ всегда
одно место слева
для знака «-»)

или "%9i"

```
printf ("%9d", x);
```

1234

всего 9 позиций

5

4

Вывод вещественных чисел

```
float x = 123.4567;  
printf ("%f", x);
```

123.456700

минимальное число
позиций, 6 цифр в
дробной части

```
printf ("%9.3f",  
x);
```

123.457

всего 9 позиций,
3 цифры в дробной
части

```
printf ("%e", x);
```

1.234560e+02

стандартный вид:
 $1,23456 \cdot 10^2$

```
printf ("%10.2e", x);
```

1.23e+02

всего 10 позиций,
2 цифры в дробной
части мантииссы

Полное решение

```
#include <stdio.h>
#include <Windows.h>
main()
{
    int a, b, c;
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);
    printf("Введите два целых числа\n");
    scanf("%d%d", &a, &b);
    c = a + b;
    printf("%d+%d=%d", a, b, c);
}
```

Протокол:

Введите два целых числа

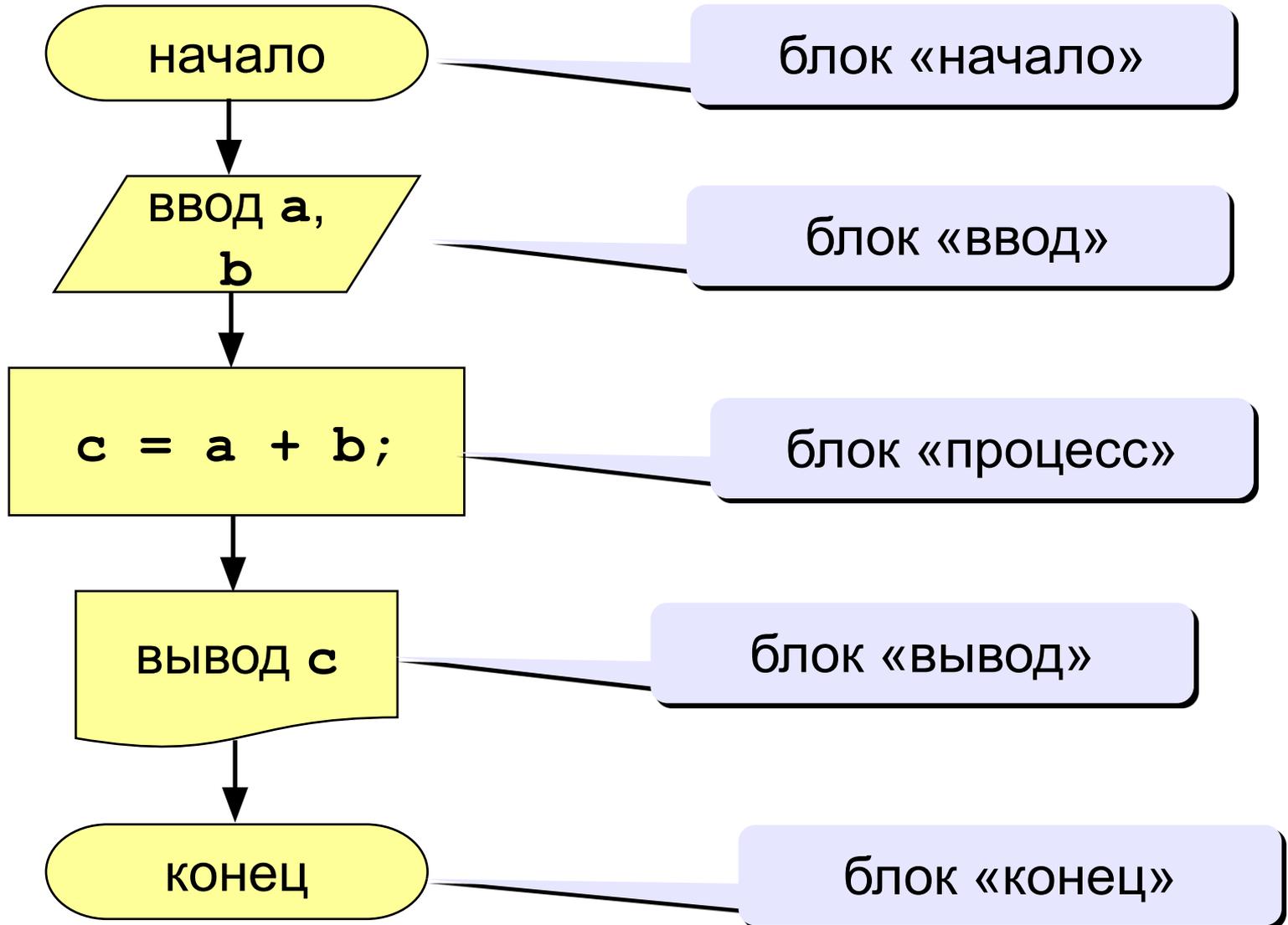
25 30

25+30=55

ЭТО ВЫВОДИТ
КОМПЬЮТЕР

ЭТО ВВОДИТ
ПОЛЬЗОВАТЕЛЬ

Линейный алгоритм



Задания

«4»: Ввести три числа, найти их сумму и произведение.

Пример:

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

«5»: Ввести три числа, найти их сумму, произведение и среднее арифметическое.

Пример:

Введите три числа:

4 5 7

$$4+5+7=16$$

$$4*5*7=140$$

$$(4+5+7) / 3 = 5.33$$

МИКРОПРОЦЕССОРН АЯ ТЕХНИКА

Практические занятия

Занятие 3



УПРАВЛЕНИЕ ИСПОЛНЕНИЕМ ПРОГРАММЫ В СИ

Алгоритмы на языке Си описываются при помощи специальных ключевых слов – **операторов управления**, определяющих последовательность выполнения действий в программе:

- Проверка и условное исполнение кода (**if-else**, «?:»)
- Переходы (**goto**)
- Циклы (**while, for, do-while**)
- Выбор (**switch, if-else if**)



Разветвляющиеся алгоритмы

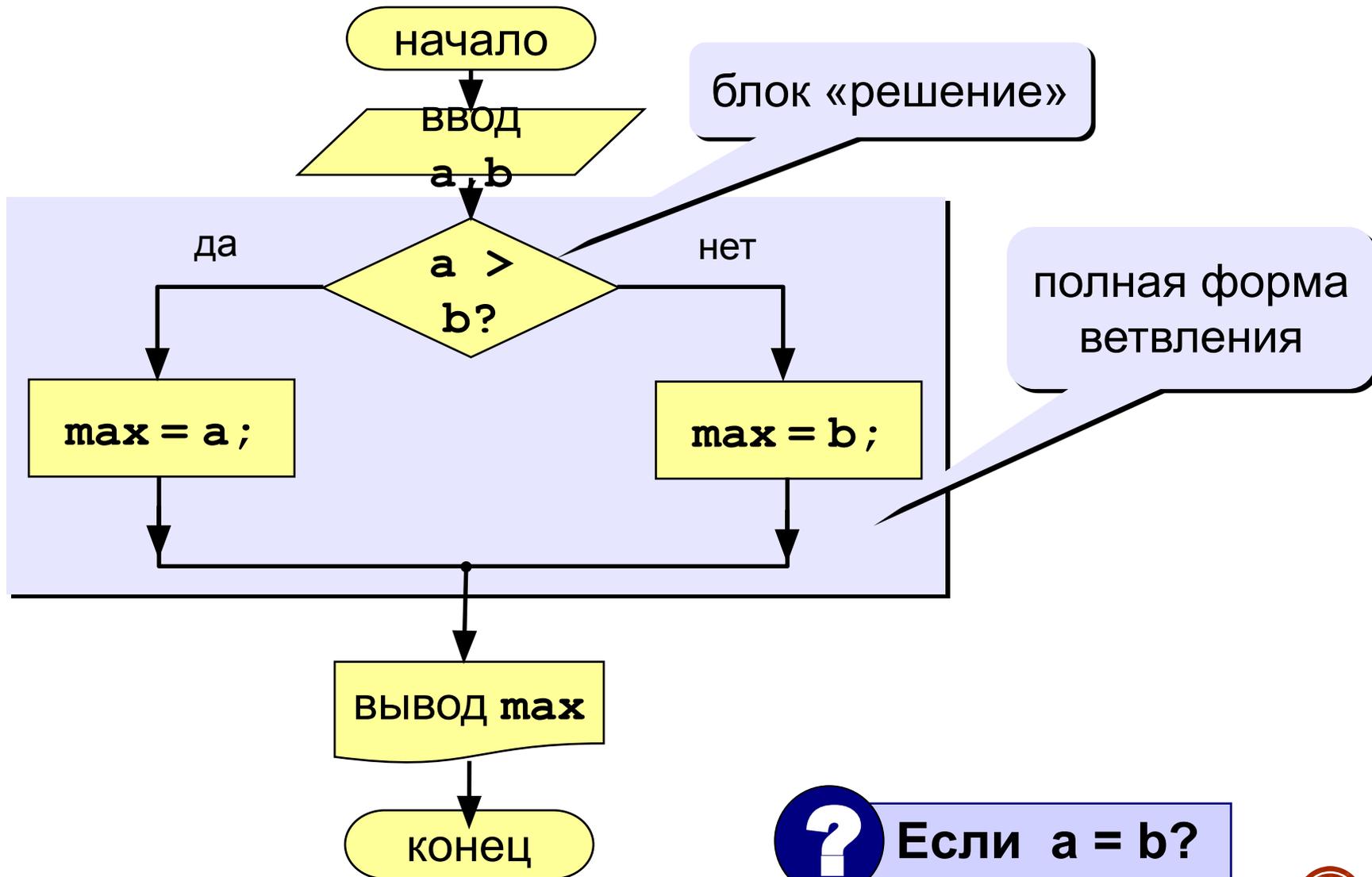
Задача. Ввести два целых числа и вывести на экран наибольшее из них.

Идея решения: надо вывести на экран первое число, если оно больше второго, или второе, если оно больше первого.

Особенность: действия исполнителя зависят от некоторых условий (*если ... иначе ...*).

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются **разветвляющимися**.

Вариант 1. Блок-схема



Вариант 1. Программа

```
main()  
{  
    int a, b, max;  
    printf("Введите два целых числа:\n");  
    scanf("%d%d", &a, &b );  
    if (a > b) {  
        max = a;  
    }  
    else {  
        max = b;  
    }  
    printf("Наибольшее число: %d", max);  
}
```

полная форма
условного
оператора

Оператор условия (условного исполнения)

```
if ( условие )
{
    // что делать, если условие верно
}
else
{
    // что делать, если условие неверно
}
```

Особенности:

- вторая часть (***else*** ...) может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать { }

Что неправильно?

```
if ( a > b )  
{  
    a = b;  
}  
else  
    b = a;
```

```
if ( a > b ) a = b;  
else  
    b = a;
```

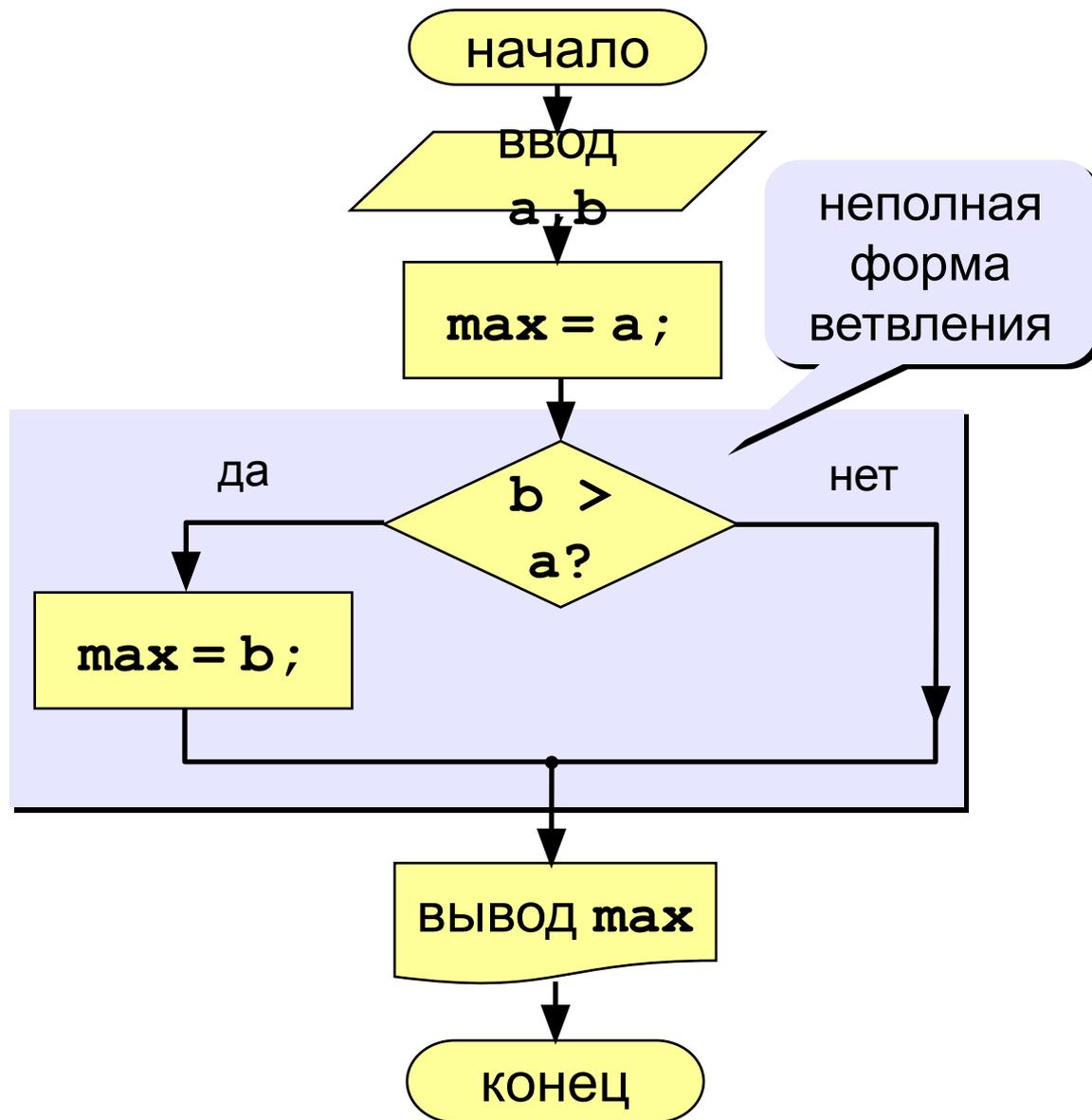
```
if ( a > b ) {  
    a = b; }  
else  
    b = a;
```

```
if ( a > b ) {  
    a = b;  
    c = 2*a; }  
else  
    b = a;
```

Обратите внимание:

фигурные скобки можно ставить как вместе с операторами, так и на отдельной строке.

Вариант 2. Блок-схема



Вариант 2. Программа

```
main()  
{  
    int a, b, max;  
    printf("Введите два целых числа:\n");  
    scanf("%d%d", &a, &b );  
    max = a;  
    if (b > a)  
        max = b;  
    printf("Наибольшее число %d", max);  
}
```

неполная форма
условного
оператора

Задача: Найти корни квадратного уравнения

▪

$$ax^2 + bx + c = 0$$

Сначала считаем дискриминант $D = b^2 - 4ac$;

если $D \geq 0$, то корни вещественные:

$$x_1 = \frac{-b + \sqrt{D}}{2a}; x_2 = \frac{-b - \sqrt{D}}{2a};$$

если $D < 0$, то корни комплексные:

$$x_1 = -\frac{b}{2a} + i \cdot \frac{\sqrt{-D}}{2a}; x_2 = -\frac{b}{2a} - i \cdot \frac{\sqrt{-D}}{2a}$$



Программа

```
main ()
```

```
{
```

```
double a, b, c, D;
```

```
printf("Введите коэффициенты квадратного уравнения:\n");
```

```
scanf("%lf%lf%lf", &a, &b, &c);
```

```
D = b*b - 4*a*c;
```

```
if (D >= 0) {
```

```
double x1 = (-b + sqrt(D)) / (2*a);
```

```
double x2 = (-b - sqrt(D)) / (2*a);
```

```
printf("Корни вещественные: x1=%lf, x2=%lf", x1, x2);
```

```
}
```

```
else {
```

```
double real = -b / (2*a);
```

```
double im = sqrt(-D) / (2*a);
```

```
printf("Корни мнимые: x1=%lf+%lfi, x2=%lf-%lfi", real, im, real, im);
```

```
}
```

```
}
```

тип всех переменных здесь – числа с плавающей точкой двойной точности **double**

для ввода и вывода чисел типа **double** используем формат **"%lf"**

для возведения в квадрат просто умножаем число само на себя

внутри фигурных скобок можно объявлять локальные переменные

для извлечения корня используется функция **sqrt()** из библиотеки **math**

чтобы напечатать одно и то же число два раза, его приходится дважды указывать в функции **printf()**

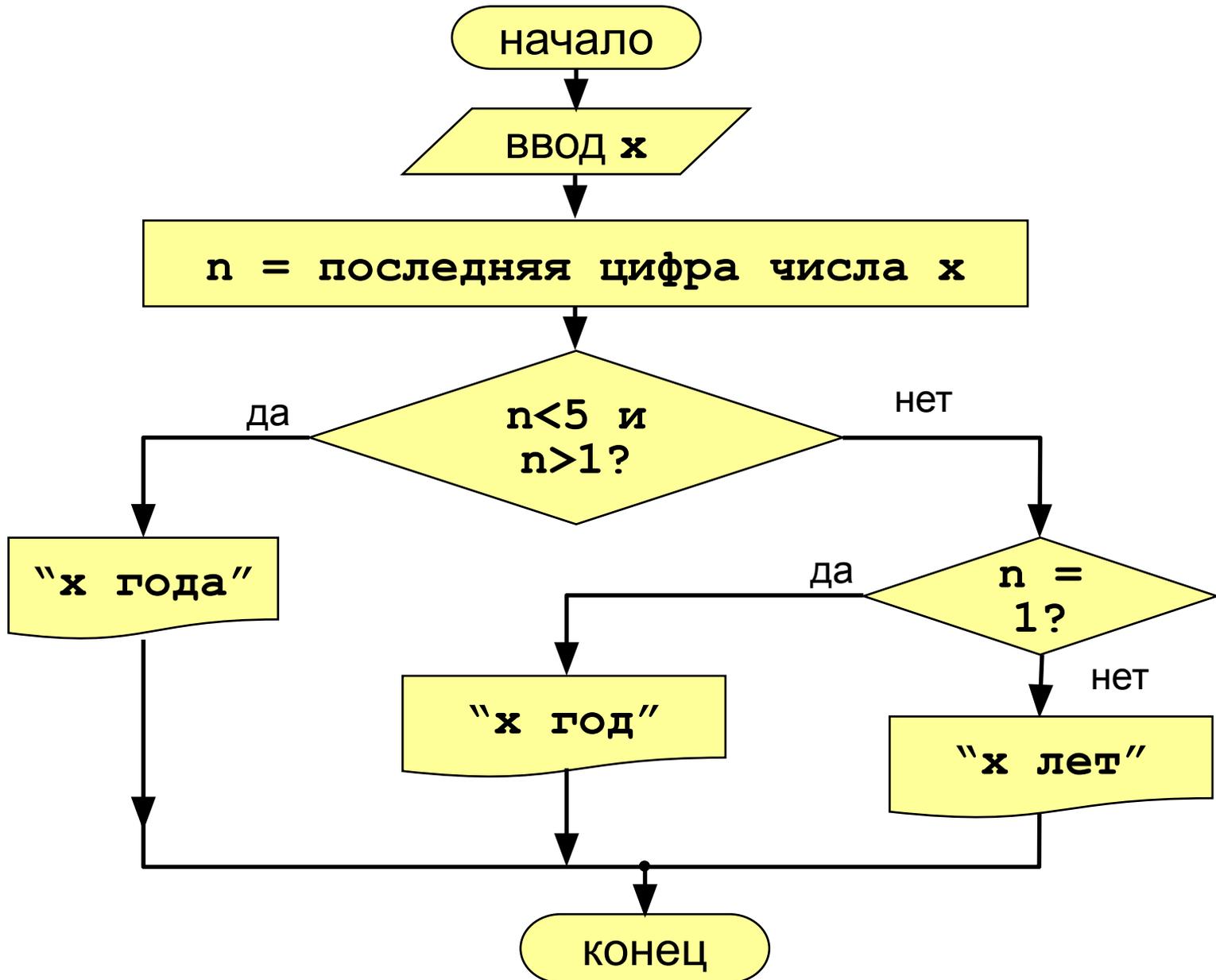
Улучшения: - Проверять, не равно ли **a** нулю
- Отдельно обрабатывать случай с **D = 0** (совпадающие корни)

Сложные условия

Задача. Надо вывести возраст человека так, чтобы после числа было напечатано слово «год», «года» или «лет».

Особенность: надо проверять, выполняются ли два условия *одновременно*. Говоря математическим языком, мы должны решить задачу на принадлежность точки к отрезку.

Алгоритм



Программа

```
main()
{
    int x;
    printf("Введите возраст\n");
    scanf("%d", &x);
    int n = x % 10;
    if ( n < 5 && n > 1 )
        printf("%d года", x);
    else if (n == 1)
        printf("%d год", x);
    else
        printf("%d лет", x);
}
```

"%" – операция
нахождения остатка
от деления

СЛОЖНОЕ
УСЛОВИЕ

Сложные условия

Сложное условие – это условие, состоящее из нескольких простых условий (отношений), связанных с помощью **логических операций**:

! – НЕ (*not*, отрицание, инверсия)

&& – И (*and*, логическое умножение, конъюнкция, одновременное выполнение условий)

|| – ИЛИ (*or*, логическое сложение, дизъюнкция, выполнение хотя бы одного из условий)

Простые условия (отношения)

<

<=

>

>=

==

!=

равно

не равно

Сложные условия

Порядок выполнения условных выражений:

- выражения в скобках
- ! (НЕ, отрицание)
- <, <=, >, >=
- ==, !=
- && (И)
- || (ИЛИ)

Пример:

```
if ( 2      1      6      3      5      4
    ! (a > b) || c != d && b == a)
{
    ...
}
```

Сложные условия

Истинно или ложно при $a=2$; $b=3$; $c=4$:

$!(a > b)$

1

$a < b \ \&\& \ b < c$

1

$!(a \geq b) \ || \ c == d$

1

$a < c \ || \ b < c \ \&\& \ b < a$

1

$a > b \ || \ !(b < c)$

0

Для каких значений **x** истинны условия:

$x < 6 \ \&\& \ x < 10$

$x < 6 \ \&\& \ x > 10$

$x > 6 \ \&\& \ x < 10$

$x > 6 \ \&\& \ x > 10$

$x < 6 \ || \ x < 10$

$x < 6 \ || \ x > 10$

$x > 6 \ || \ x < 10$

$x > 6 \ || \ x > 10$

| | |
|----------------------------------|----------|
| $(-\infty, 6)$ | $x < 6$ |
| \emptyset | |
| $(6, 10)$ | |
| $(10, \infty)$ | $x > 10$ |
| $(-\infty, 10)$ | $x < 10$ |
| $(-\infty, 6) \cup (10, \infty)$ | |
| $(-\infty, \infty)$ | |
| $(6, \infty)$ | $x > 6$ |

Задания

«4»: Ввести пять чисел и найти наибольшее из них.

Пример:

Введите пять чисел:

4 15 9 56 4

Наибольшее число 56

«5»: Ввести номер месяца и вывести название времени года.

Пример:

Введите номер месяца:

4

весна

МИКРОПРОЦЕССОРН АЯ ТЕХНИКА

Практические занятия

Занятие 4



Циклы

Цикл – это многократное выполнение одинаковой последовательности действий.

- цикл с **известным** числом шагов (цикл со счётчиком)
- цикл с **неизвестным** числом шагов (итерационный цикл)

Цикл с неизвестным числом шагов

Пример: Отпилить полено от бревна. Сколько раз надо сделать движения пилой?

Задача: Ввести целое число (<2000000) и определить число цифр в нем.

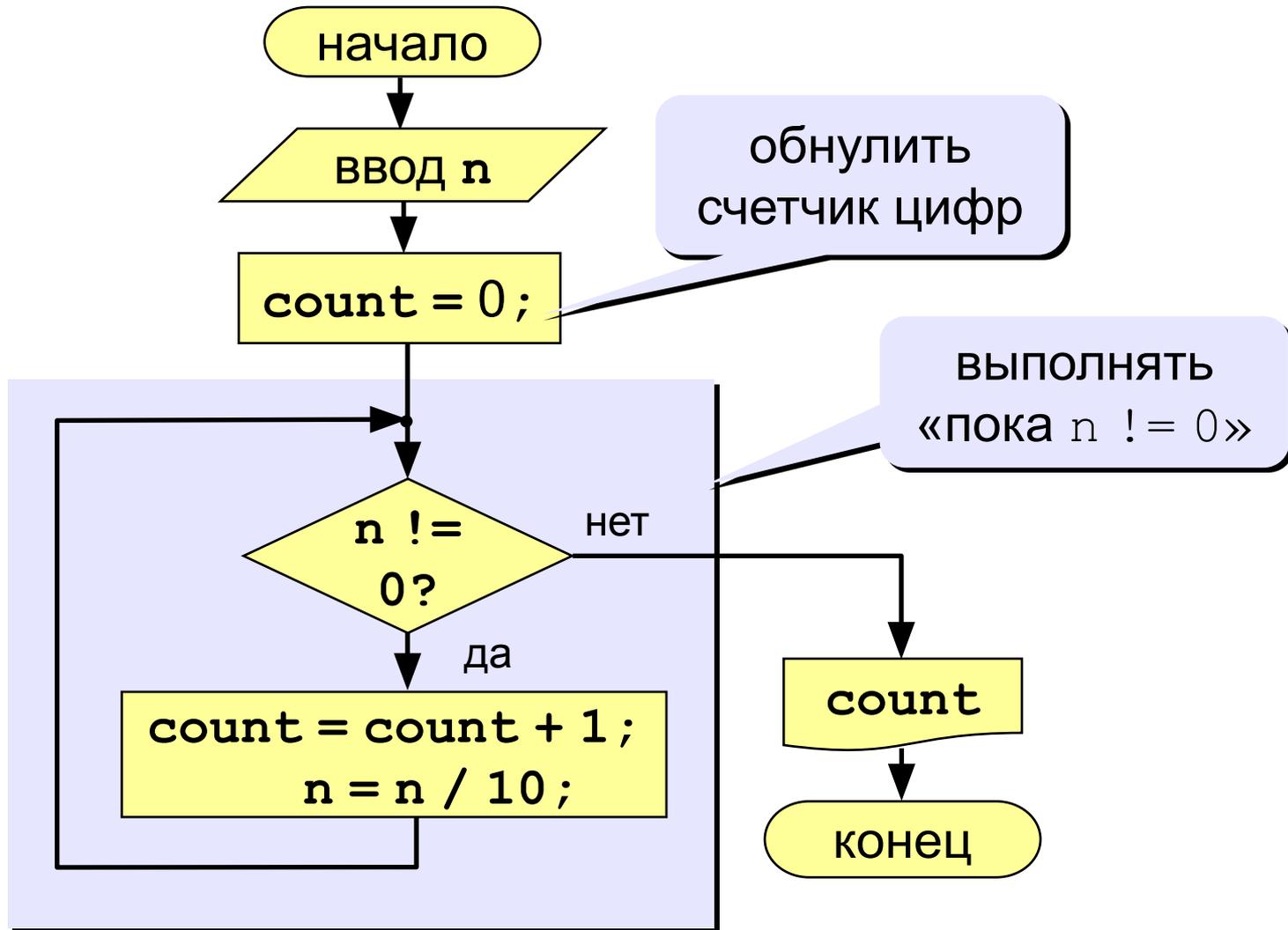
Идея решения: Отсекаем последовательно последнюю цифру (путем деления числа на 10), увеличиваем счетчик.

| n | count |
|-----|-------|
| 123 | 0 |
| 12 | 1 |
| 1 | 2 |
| 0 | 3 |

Проблема: Неизвестно, сколько шагов надо сделать.

Решение: Надо остановиться, когда $n = 0$.

Алгоритм



Программа

```
main()
{
  int n, count, n1;
  printf("Введите целое число\n");
  scanf("%d", &n);
  count = 0; n1 = n;
  while (n != 0)
  {
    count++;
    n = n / 10;
  }
  printf("В числе %d нашли %d цифр", n1,
        count);
}
```

ВЫПОЛНЯТЬ
«ПОКА n != 0»

 Что плохо?

Цикл с условием

```
while ( условие )  
  {  
    // тело цикла  
  }
```

Особенности:

- МОЖНО ИСПОЛЬЗОВАТЬ СЛОЖНЫЕ УСЛОВИЯ:

```
while ( a < b && b < c ) { ... }
```

- если в теле цикла только один оператор, скобки **{ }** можно не писать:

```
while ( a < b ) a ++;
```

Цикл с условием

Особенности:

- условие проверяется **каждый раз** при входе в цикл
- если условие на входе в цикл ложно, цикл не выполняется ни разу

```
a = 4; b = 6;  
while ( a > b ) a = a - b;
```

- если условие никогда не станет ложным, программа **зацикливается**

```
while (1)  
{  
// главный цикл программы  
}
```

Сколько раз выполняется цикл?

```
a = 4; b = 6;  
while ( a < b ) a ++;
```

2 раза

a = 6

```
a = 4; b = 6;  
while ( a < b ) a += b;
```

1 раз

a = 10

```
a = 4; b = 6;  
while ( a > b ) a ++;
```

0 раз

a = 4

```
a = 4; b = 6;  
while ( a < b ) b = a - b;
```

1 раз

b = -2

```
a = 4; b = 6;  
while ( a < b ) a --;
```

зацикливание

Задания

«4»: Ввести целое число и найти сумму его цифр.

Пример:

Введите целое число:

1234

Сумма цифр числа 1234 равна 10.

«5»: Ввести целое число и определить, верно ли, что в его записи есть две одинаковые цифры.

Пример:

Введите целое число:

1234

Нет.

Введите целое число:

1224

Да.

Цикл с постусловием

Задача: Ввести целое **положительное** число (<2000000) и определить число цифр в нем.

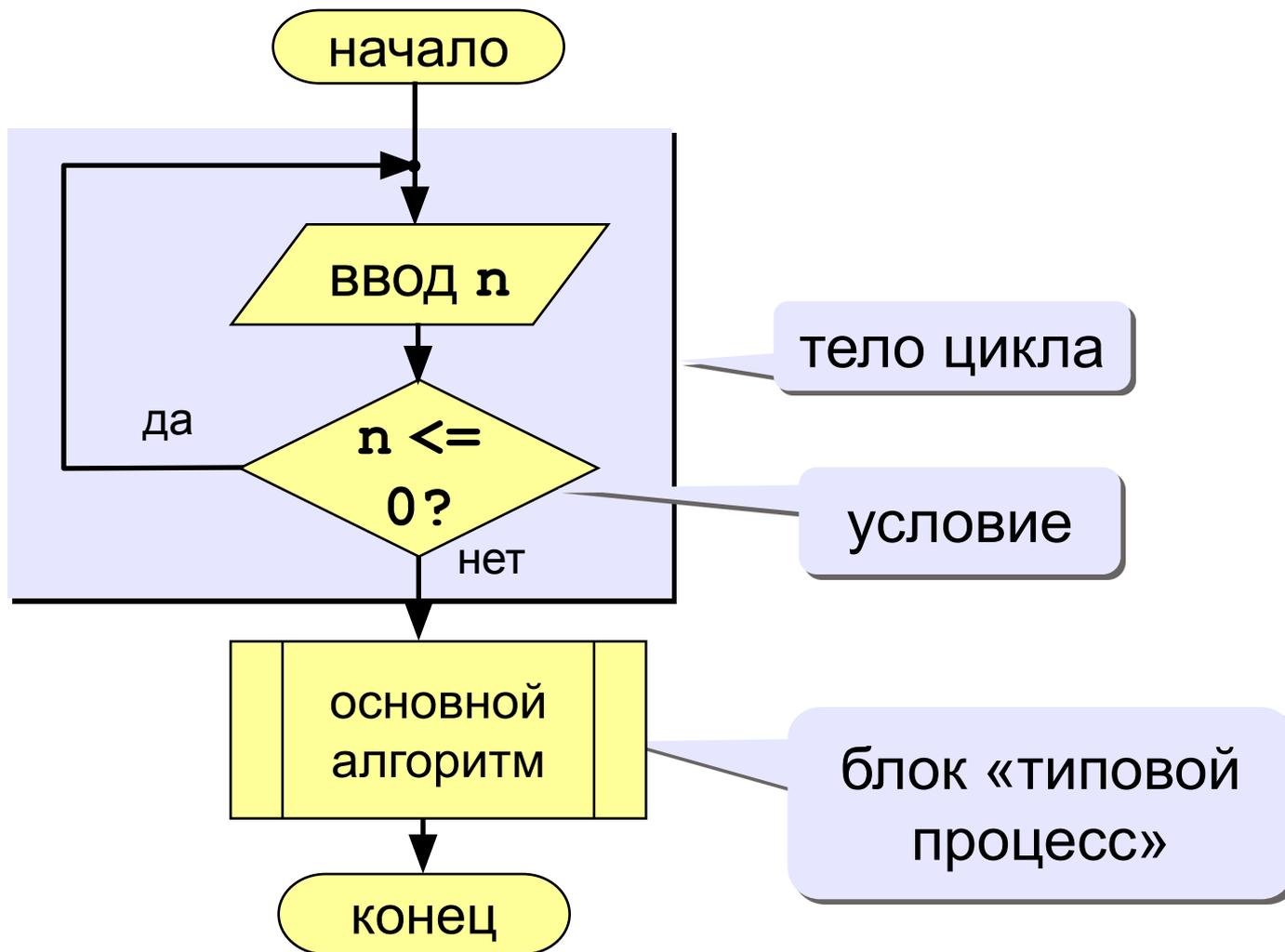
Проблема: Как не дать ввести отрицательное число или ноль?

Решение: Если вводится неверное число, вернуться назад к вводу данных (цикл!).

Особенность: Один раз тело цикла надо сделать в любом случае \Rightarrow проверку условия цикла надо делать в конце цикла (цикл с **постусловием**).

Цикл с постусловием – это цикл, в котором проверка условия выполняется в конце цикла.

Цикл с постусловием: алгоритм



Программа

```
main()
{
    int n;
    do {
        printf("Введите положительное число\n");
        scanf("%d", &n);
    }
    while ( n <= 0 );
    ... // основной алгоритм
}
```

условие

Особенности:

- тело цикла всегда выполняется **хотя бы один раз**
- после слова **while** («пока...») ставится условие **продолжения** цикла

Сколько раз выполняется цикл?

```
a = 4; b = 6;  
do { a++; } while (a <= b);
```

3 раза

a = 7

```
a = 4; b = 6;  
do { a += b; } while (a <= b);
```

1 раз

a = 10

```
a = 4; b = 6;  
do { a += b; } while (a >= b);
```

зацикливание

```
a = 4; b = 6;  
do b = a - b; while (a >= b);
```

2 раза

b = 6

```
a = 4; b = 6;  
do a += 2; while (a >= b);
```

зацикливание

Задания (с защитой от неверного ввода)

«4»: Ввести натуральное число и определить, верно ли, что сумма его цифр равна 10.

Пример:

Введите число ≥ 0 :

-234

Нужно положительное число. Нет

Введите число ≥ 0 :

1234

Да

Введите число ≥ 0 :

1233

Нет

«5»: Ввести натуральное число и определить, какие цифры встречаются несколько раз.

Пример:

Введите число ≥ 0 :

2323

Повторяются: 2, 3

Введите число ≥ 0 :

1234

Нет повторов.

МИКРОПРОЦЕССОРН АЯ ТЕХНИКА

Практические занятия

Занятие 5



Цикл с известным числом шагов

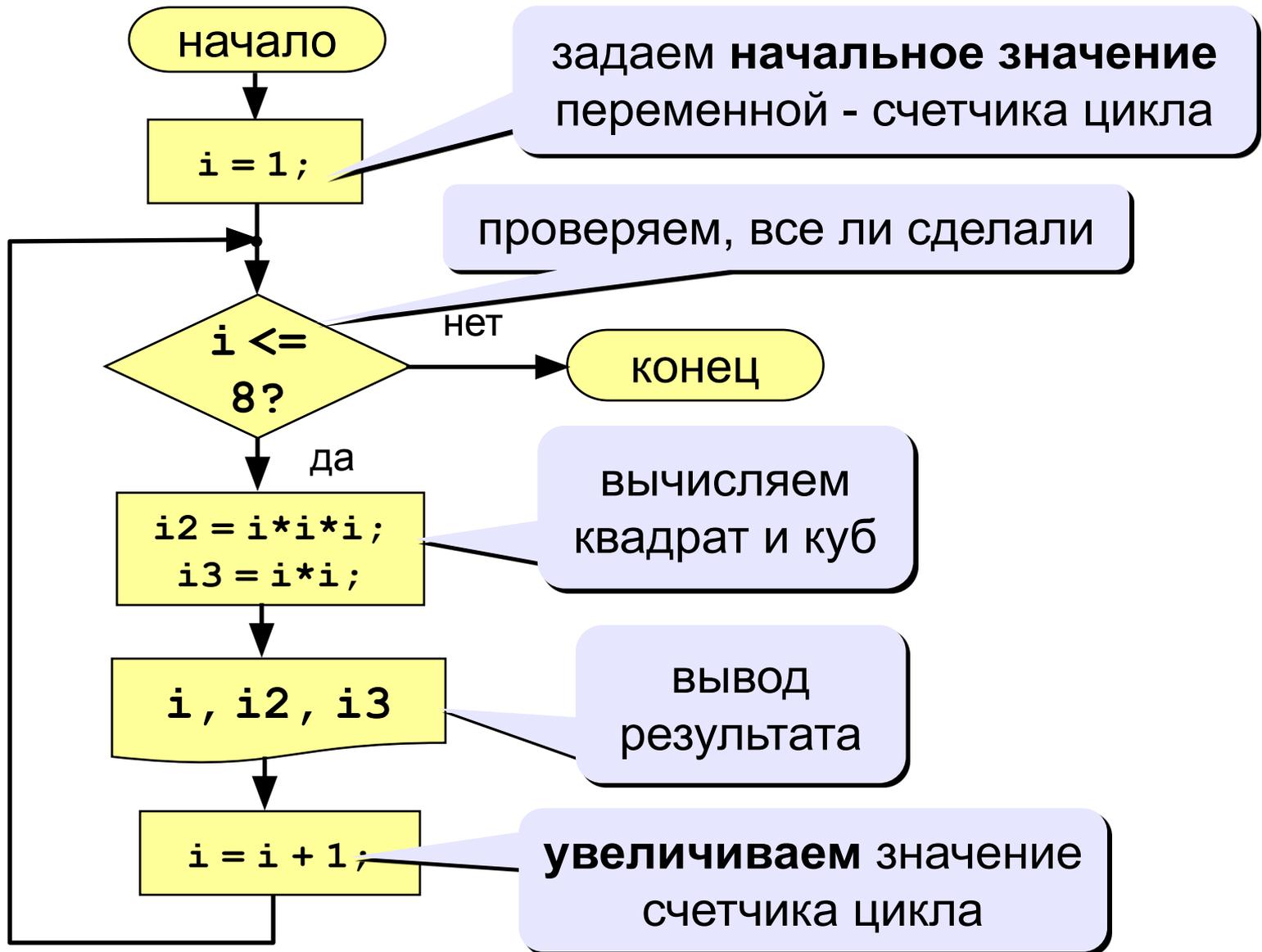
Цикл с известным числом шагов организуется при помощи специальной переменной – *счетчика цикла*.

Задача. Вывести на экран квадраты и кубы целых чисел от 1 до 8 (от **a** до **b**).

Особенность: одинаковые действия выполняются 8 раз.



Алгоритм



Программа

```
main ()
```

```
{
```

```
int i = 0;
```

Объявляем переменную -
счетчик цикла и присваиваем ей
начальное значение

```
while (i <= 8)
```

Проверяем условие выхода из
цикла

```
{
```

```
int i2 = i*i;
```

Значение переменной-счетчика
можно использовать в теле цикла

```
int i3 = i*i*i;
```

```
printf ("%d %d %d\n", i, i2, i3);
```

```
i++;
```

увеличиваем значение
счетчика цикла

```
}
```

```
}
```

Проблема:

Действия со счетчиком цикла разбросаны по тексту →
велика вероятность где-нибудь ошибиться!

Оператор цикла for

- Оператор **for** используется для *краткого описания* циклов с заранее известным числом повторений
- В операторе **for** переменной-счетчику:
 - присваивают начальное значение;
 - задают условие завершения цикла;
 - указывают операцию по изменению значения счетчика в конце каждого шага.

```
int i = 0;
while (i <= 8)
{
    printf("%d %d %d\n", i, i*i, i*i*i);
    i++;
}
```

```
int i;
for (i = 0; i <= 8; i++)
{
    printf("%d %d %d\n", i, i*i, i*i*i);
}
```

Цикл с уменьшением счетчика

Задача. Вывести на экран квадраты и кубы целых чисел от 8 до 1 (в обратном порядке).

Особенность: переменная цикла должна *уменьшаться*.

Решение:

```
for ( i = 8; i >= 1; i -- )  
{  
    printf("%4d %4d %4d\n", i, i*i, i*i*i);  
}
```

сделаем заодно ровные
столбики

Оператор цикла `for` – общий вид

```
for (начальные значения;  
     условие продолжения цикла;  
     действия на каждом шаге)  
{  
    // тело цикла  
}
```

Примеры:

```
for (a = 2; a < b; a += 2) { ... }
```

```
for (a = 2, b = 4; a < b; a += 2) { ... }
```

```
for (a = 1; c < d; x++) { ... }
```

```
for (; c < d; x++) { ... }
```

```
for (; c < d; ) { ... }
```

Оператор цикла `for`

Особенности:

- **условие** проверяется **в начале** очередного шага цикла, если оно ложно, то цикл завершается;
- **действия** (третья часть в заголовке) выполняются **в конце** очередного шага цикла;
- если **условие** никогда не станет ложным, цикл может продолжаться бесконечно (**зацикливание**)

```
for (i=1; i<8; i++) { i--; }
```



Не рекомендуется менять счетчик цикла внутри тела цикла!

- если в теле цикла есть только один оператор, скобки `{ }` можно не ставить:

```
for (i=1; i<8; i++) a += b;
```

Сколько раз выполняется цикл?

```
a = 1;  
for (i=1; i<4; i++) a++;
```

a = 4

```
a = 1;  
for (i=1; i<4; i++) a = a+i;
```

a = 7

```
a = 1; b=2;  
for (i=3; i >= 1; i--) a += b;
```

a = 7

```
a = 1;  
for (i=1; i >= 3; i--) a = a+1;
```

a = 1

```
a = 1;  
for (i=1; i <= 4; i--) a ++;
```

зацикливание

Замена for на while и наоборот

```
for ( i=1; i<=10; i++)  
{  
    // тело цикла  
}
```

```
i = 1;  
while ( i <= 10 ) {  
    // тело цикла  
    i ++;  
}
```

```
for ( i=a; i>=b; i-- )  
{  
    // тело цикла  
}
```

```
i = a;  
while ( i >= b ) {  
    // тело цикла  
    i --;  
}
```



В языке Си замена цикла *for* на *while* и наоборот возможна **всегда!**

Задания

«4»: Ввести a и b и вывести квадраты и кубы чисел от a до b .

Пример:

Введите границы интервала :

4 6

4 16 64

5 25 125

6 36 216

«5»: Вывести квадраты и кубы 10 чисел следующей последовательности: 1, 2, 4, 7, 11, 16, ...

Пример:

1 1 1

2 4 8

4 16 64

...

46 2116 97336

Оператор выбора

Задача: Ввести номер месяца и вывести количество дней в этом месяце.

Решение: Число дней по месяцам:

28 дней – 2 (февраль)

30 дней – 4 (апрель), 6 (июнь), 9 (сентябрь), 11 (ноябрь)

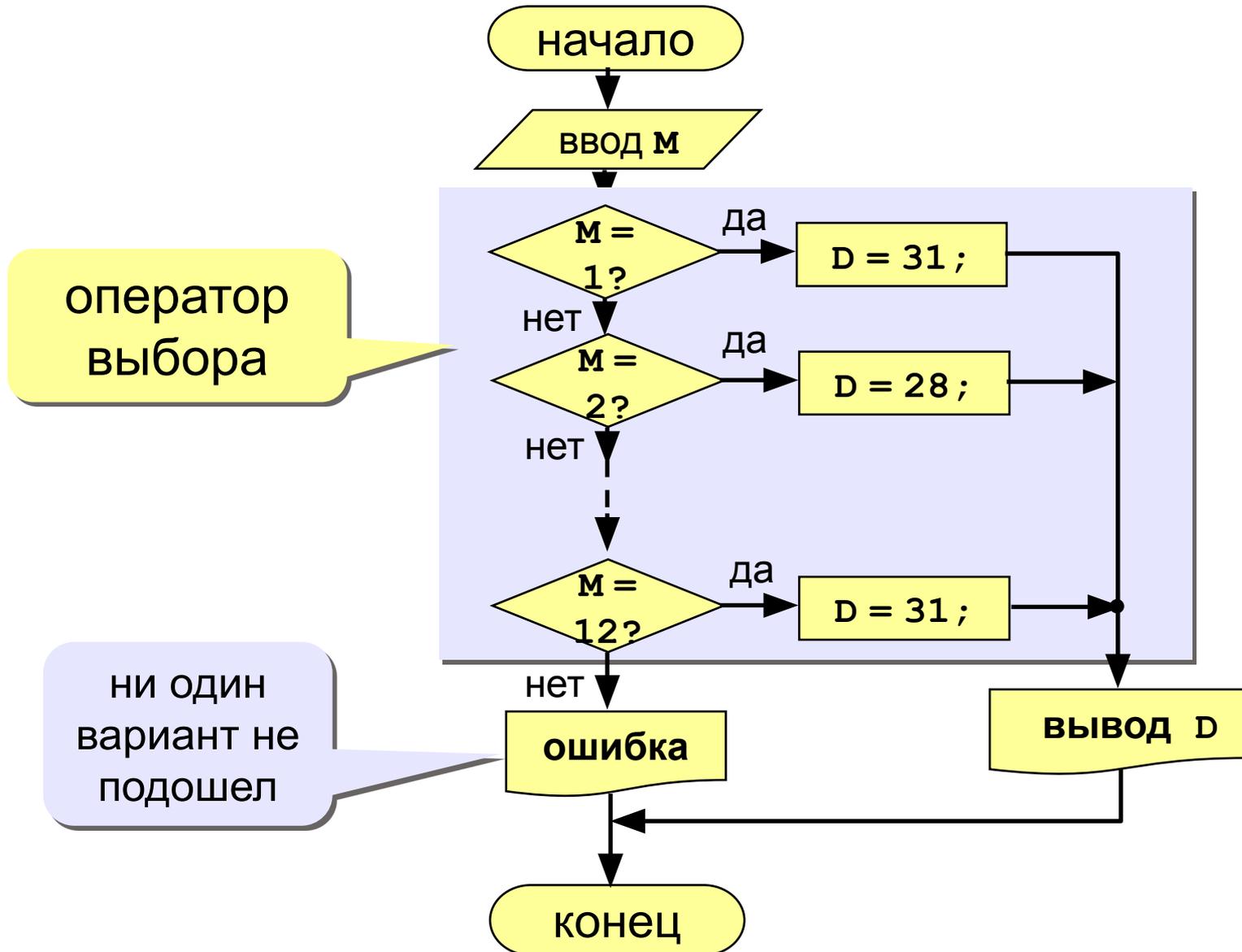
31 день – 1 (январь), 3 (март), 5 (май), 7 (июль),
8 (август), 10 (октябрь), 12 (декабрь)

Особенность: Выбор не из двух, а из нескольких вариантов в зависимости от номера месяца.



Можно ли решить известными методами?

Алгоритм



Программа

```
main()
{
    int M, D;
    printf("Введите номер месяца:\n");
    scanf("%d", &M);

    switch ( M ) {
        case 2:  D = 28; break;
        case 4: case 6: case 9: case 11:
                D = 30; break;
        case 1: case 3: case 5: case 7:
        case 8: case 10: case 12:
                D = 31; break;
        default: D = -1;
    }

    if (D > 0)
        printf("В этом месяце %d дней.", D);
    else printf("Неверный номер месяца");
}
```

ВЫЙТИ ИЗ
switch

НИ ОДИН
вариант не
подошел

Оператор выбора

Задача: Ввести букву и вывести название животного на эту букву.

Особенность: выбор по символьной величине.

```
main()
{
    char c;
    printf("Введите первую букву названия животного:\n");
    scanf("%c", &c);

    switch ( c ) {
        case 'a': printf("Антилопа"); break;
        case 'б': printf("Бизон"); break;
        case 'в': printf("Волк"); break;
        default:  printf("Я не знаю!");
    }
}
```



Что будет, если везде убрать break?

Оператор выбора `switch`

Особенности:

- после **`switch`** может быть имя переменной или арифметическое выражение целого типа (**`int`**) или символьного типа (**`char`**)

```
switch ( i+3 ) {  
    case 1: a = b; break;  
    case 2: a = c;  
}
```

- **нельзя** ставить два **одинаковых** значения:

```
switch ( x ) {  
    case 1: a = b; break;  
    case 1: a = c;  
}
```

Задание

Ввести номер месяца и номер дня, вывести число дней, оставшихся до Нового года.

Пример:

Введите номер месяца:

12

Введите день:

25

До Нового года осталось 6 дней.

МИКРОПРОЦЕССОРН АЯ ТЕХНИКА

Практические занятия

Занятие 6



Ввод/вывод СИМВОЛОВ

- Самый простой механизм ввода – чтение по одному символу из стандартного входного потока (с клавиатуры) и вывод по одному символу в стандартный выходной поток (на экран дисплея, в консольное окно).
- Функция **getch()** ожидает нажатие клавиши и возвращает ее код:

```
int key = getch();
```

- Функция **putch(int x)** выдает символ с кодом x в стандартный выходной поток (в консольное окно):

```
putch('H'); putch('i'); putch('!');
```



Hi!

- Функции **getch()** и **putch()**, так же, как и **printf()**, **scanf()**, объявлены в заголовочном файле **stdio.h**.



Ввод/вывод СИМВОЛОВ

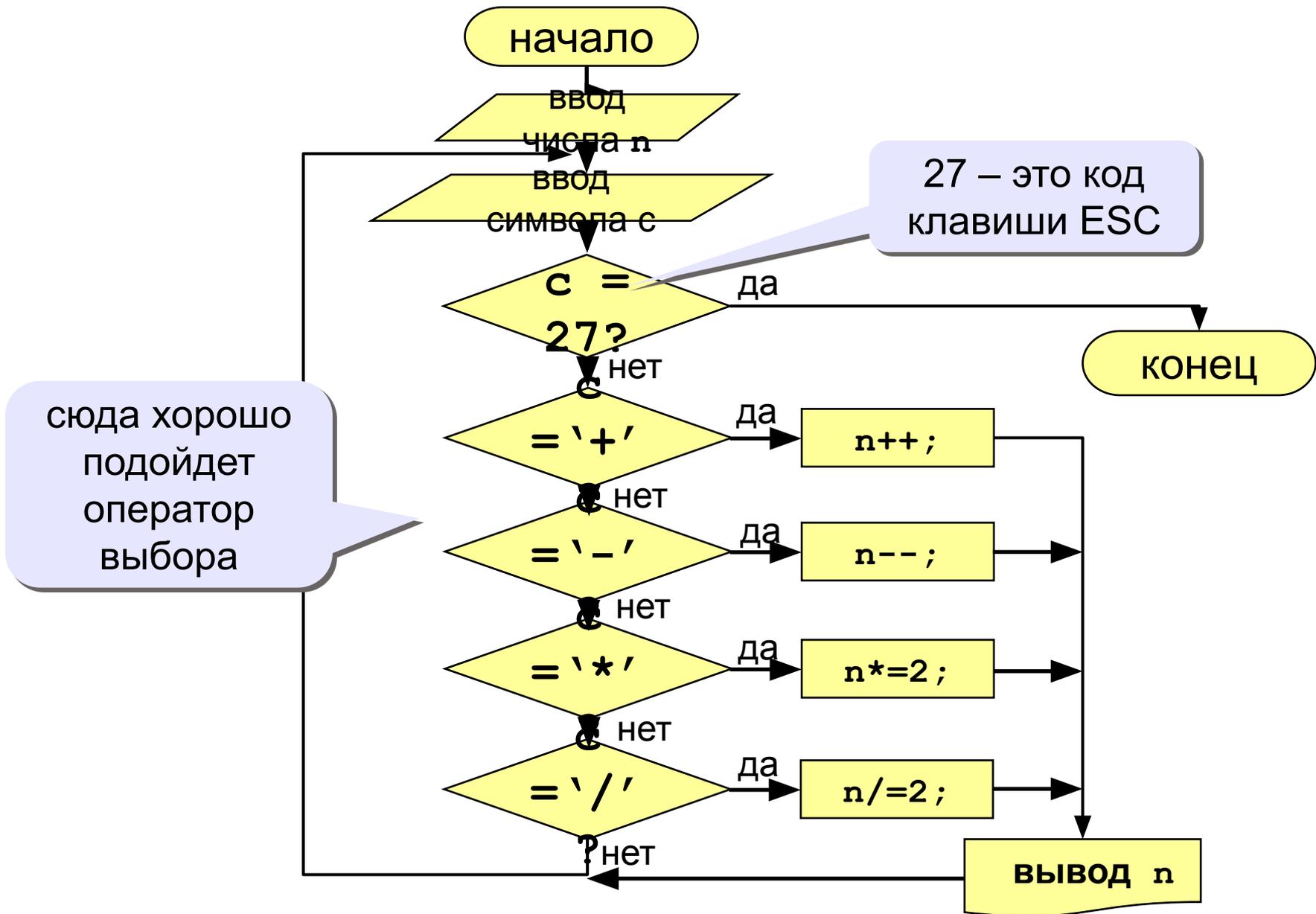
Задача. Ввести число. Затем обеспечить следующую функциональность:

-  клавиши «+» и «-» увеличивают / уменьшают число на 1;
-  клавиши «*» и «/» увеличивают / уменьшают число в два раза;
-  клавиша «ESC» вызывает выход из программы.

Особенность: алгоритм представляет собою цикл с заранее неизвестным числом повторений.



Ввод/вывод символов: алгоритм



Программа

```
main ()
{
  int n, c;
  printf("Введите число:\n");
  scanf("%d", &n);
  do {
    c = getch();
    switch ( c )
    {
      case '+': n++; break;
      case '-': n--; break;
      case '*': n*=2; break;
      case '/': n/=2; break;
    }
    printf("%9d\r", n);
  } while (c != 27);
}
```

останавливаем
программу, читаем
код клавиши

ВМЕСТО КОДА СИМВОЛА
МОЖНО НАПИСАТЬ САМ
ЭТОТ СИМВОЛ ВНУТРИ
ОДИНАРНЫХ КАВЫЧЕК ' '

"\r" вызывает
печать с
начала строки

Меню в программе

Задача. Написать программу, вычисляющую площади различных фигур. Тип фигуры и ее параметры должны вводиться с клавиатуры.

Особенность: Подобного рода программы обычно содержат «меню пользователя», в котором можно сделать выбор, нажав одну из предложенных кнопок.



Программа

```
main ()
{
    printf ("Вычисление площадей фигур:\n");
    printf ("1: Круг\n");
    printf ("2: Квадрат\n");
    printf ("3: Треугольник\n");
    printf ("0: Выход из программы\n");
    int c = getch ();
    double result;
    switch ( c )
    {
        case 0: return;
        case 1: result = calc_round (); break;
        case 2: result = calc_square (); break;
        case 3: result = calc_triangle (); break;
    }
    printf ("Площадь фигуры = %lf\r", result);
}
```

просто выходим из программы, даже не завершая switch()

вызываем разные функции для вычислений

Задания

- «4»:** Ввести два числа, затем сделать так, чтобы нажатие «+» увеличивало бы первое число на величину, заданную вторым числом; нажатие «-» - уменьшало бы это число на ту же величину.
- «5»:** Ввести одно число, а затем сделать так, чтобы нажатие на цифровые клавиши 1..9 увеличивало (или уменьшало) бы это число на значение, заданное нажатой клавишей.

Функции

В языке Си **функция** – группа из одного или нескольких операторов, имеющая имя. Как правило, функция выполняет одно действие, один шаг алгоритма или одно вычисление по формуле.

Функция может получать на вход данные (они размещаются в специальных переменных – *параметрах функции*) и может отдавать в основную программу результат (одно число).

Функция – основная единица программы на языке Си; собственно программа и состоит из описания функций.



Функции

Задача: составить функцию, которая вычисляет наибольшее из двух значений, и привести пример ее использования

Функция:

тип
результата

параметры
функции

оператор `return`
возвращает
результат функции

```
int Max ( int a, int b )  
{  
    if ( a > b ) return a ;  
    else      return b ;  
}
```

Функции

- Объявление функции:

```
тип_функции  имя_функции  (список_параметров)
{
    тело_функции
}
```

- Имя типа, стоящее перед именем функции, задает тип возвращаемого функцией значения. Если функция не возвращает никакого значения, в качестве типа пишут ключевое слово **void**.
- Список параметров – это список локальных переменных, автоматически получающих значения в момент вызова функции. Если функция не использует параметров, то в круглых скобках не пишут ничего или пишут слово **void**.
- Оператор **return** заканчивает выполнение функции. Если функция возвращает число, после слова **return** необходимо указать выражение, значение которого передается в качестве результата. «return» может использоваться в теле функции сколько угодно раз.

Вызов функции

Когда нужно, чтобы выполнились действия, описанные в функции, функцию *вызывают*.

Для этого есть **два способа**:

- **Если функция ничего не возвращает (у нее тип `void`):** просто написать в тексте программы имя функции, а затем, в круглых скобках, значения, которые будут присвоены параметрам функции.
- **Если функция возвращает значение:** использовать имя функции внутри какого-нибудь выражения.

Вызвать функцию можно из любой другой функции.



Вызов функции

```
int Max (int a, int b)
```

```
{  
  if (a > b)  
    return a;  
  else  
    return b;  
}
```

a=10

b=15

Вызов **Max()** из выражения в операторе присваивания

```
void main()
```

```
{  
  int x = Max (10, 15);  
  int y = Max (5, x) * 4;  
  int m = -6, n = 10;  
  printf ("Maximum = %d", Max (m, n));  
}
```

x = 15

y = 60

Вызов **Max()** из выражения в функции **printf()**



Объявление переменных и область видимости

- Язык Си позволяет объявлять переменные в любом месте файла с исходным текстом, в том числе внутри любой функции (и вообще внутри любого составного оператора).
- Переменные, объявленные снаружи любого составного оператора, доступны во всех функциях, описанных в данном файле. Такие переменные называют *глобальными*.
- Переменные, объявленные в функции, доступны только изнутри этой функции. Такие переменные называются *локальными*. Они создаются всякий раз при вызове функции и уничтожаются при выходе из нее.

```
short x, y;  
short GetAverage (short a, short b)  
{  
    return (a + b) / 2;  
}  
int main(void)  
{  
    x = 10; y = 12;  
    short z = GetAverage(x, y);  
    printf("z=%d", z);  
}
```

Параметры функции также являются ее локальными переменными. Их изменение никак не влияет на объекты, находящиеся за ее пределами.

ФУНКЦИИ

Вернемся к нашей задаче вычисления площадей фигур. Нам нужно описать три функции: для вычисления площади круга, прямоугольника и треугольника.

Особенность: Каждая такая функция будет не только производить вычисления, но и обеспечивать ввод необходимых исходных данных.

```
double calc_square(void)
{
    double r;
    printf("Введите сторону
квадрата:");
    scanf("%lf", &r);
    return r * r;
}
```

```
double calc_round (void)
{
    double r;
    printf("Введите радиус
круга:");
    scanf("%lf", &r);
    return M_PI * r * r;
}
```

Константа M_PI (число π)
объявлена в библиотеке math.h



Задания

- «4»:** Написать функцию для вычисления тока в резисторе. В качестве параметров в функцию отдаются значения напряжения на выводах резистора и его сопротивление. В основной программе обеспечить ввод исходных данных, вычисление тока при помощи написанной функции и печать результата.
- «5»:** Написать программу для вычисления корня функции методом половинного деления. Сама функция должна получать на вход значение аргумента, возвращать значение функции, для которой ведется поиск корней. В основной программе надо обеспечить ввод значений границ области поиска и требуемой точности вычисления, а также печать результатов поиска корня.