

**Курс «Алгоритмы и алгоритмические языки»
1 семестр 2013/2014**

Лекция 6

Форматный ввод-вывод

```
#include <stdio.h>  
  
int main (void)  
{  
    int s = 0;  
    int a, b;  
  
    scanf ("%d%d", &a, &b);  
    s += a + b;  
    printf ("Сумма равна %d\n", s);  
    return 0;  
}
```

Форматный ввод-вывод

Спецификаторы ввода-вывода

%d, %ld, %lld – напечатать/считать число типа `int`, `long`, `long long`

%u, %lu, %llu – напечатать/считать число типа `unsigned`, `unsigned long`, `unsigned long long`

%f, %Lf – напечатать число типа `double`, `long double`

%f, %lf, %Lf – считать число типа `float`, `double`, `long double`

%c – напечатать/считать символ

%4d – вывести число типа `int` минимум в четыре символа

%.5f – вывести число типа `double` с пятью знаками

%% – напечатать знак процента

Функция **scanf** возвращает количество удачно считанных элементов

Пример Си-программы

```
/* Решение квадратного уравнения */
#include <stdio.h>
#include <math.h>
int main (void)
{
    int a, b, c, d;
    /* Введем коэффициенты */
    scanf ("%d%d%d", &a, &b, &c);
    if (!a) {
        printf ("Уравнение не квадратное!\n");
        return 1;
    }
    d = b*b - 4*a*c;
    if (d < 0)
        printf ("Решений нет\n");
    else if (d == 0) {
        double db = -b;
        printf ("Решение: %.4f\n", db/(2*a));
    } else {
        double db = -b;
        double dd = sqrt (d);
        printf ("Решение 1: %.4f, решение 2: %.4f\n", (db+dd)/(2*a), (db-dd)/(2*a));
    }
    return 0;
}
```

Пример Си-программы

```
/* Решение квадратного уравнения */
#include <stdio.h>
#include <math.h>
int main (void)
{
    int a, b, c, d;
    /* Введем коэффициенты */
    scanf ("%d%d%d", &a, &b, &c);
    if (!a) {
        printf ("Уравнение не квадратное!\n");
        return 1;
    }
    d = b*b - 4*a*c;
```

Пример Си-программы

```
if (d < 0)
    printf ("Решений нет\n");
else if (d == 0) {
    double db = -b;
    printf ("Решение: %.4f\n", db/(2*a));
} else {
    double db = -b;
    double dd = sqrt (d);
    printf ("Решение 1: %.4f, решение 2: %.4f\n",
            (db+dd)/(2*a), (db-dd)/(2*a));
}
return 0;
}
```

Преобразование типов

- ◆ ***При присваивании: a = b***
 - ◆ “Широкий” целочисленный тип в “узкий”: отсекаются старшие биты
 - ◆ Знаковый тип в беззнаковый: знаковый бит “становится” значащим

```
signed char c = -1; /* sizeof(c) == 1 */  
((unsigned char) c) -> 255
```
 - ◆ Плавающий тип в целочисленный: отбрасывается дробная часть
 - ◆ “Широкий” плавающий тип в “узкий”: округление или усечение числа

- ◆ ***Явное приведение типов: (type) expression***
 - ◆ Пример `d = ((double) a+b)/2;`

Приведение типов



Неявное приведение типов: происходит, когда операнды двухместной операции имеют разные типы

- ◆ Если один из операндов – `long double`, то и второй преобразуется к `long double` (так же для `double` и `float`)
`long double + double -> long double + long double`
`int + double -> double + double`
`float + short -> float + int -> float + float`
- ◆ Если все значения операнда могут быть представлены в `int`, то операнд преобразуется к `int`, так же и для `unsigned int` (англоязычный термин – `integer promotion`)
`unsigned short(2) + char(1) -> int(4) + int(4)`
`unsigned short(4) + char(1) -> unsigned int(4) + int(4)`
- ◆ Если оба операнда – соответственно знаковых или беззнаковых целых типов, то операнд более “узкого” типа преобразуется к операнду более “широкого” типа
`int + long -> long + long`
`unsigned long long + unsigned ->`
`unsigned long long + unsigned long long`

Приведение типов



Неявное приведение типов: происходит, когда операнды двухместной операции имеют разные типы

- ◆ Если операнд беззнакового типа имеет более “широкий” тип, чем операнд знакового “узкого” типа, то операнд “узкого” типа преобразуется к операнду “широкого” типа
`int + unsigned long -> unsigned long + unsigned long`

`int(4) / unsigned int(4) -> unsigned int(4) / unsigned int(4) /* Неверные значения */`

- ◆ Если тип операнда знакового типа может представить все значения типа операнда беззнакового типа, то операнд беззнакового типа преобразуется к операнду знакового типа

`unsigned int(4) + long(8) -> long(8) + long(8)`

`unsigned short + long long -> long long + long long`

- ◆ Оба операнда преобразуются к беззнаковому типу, соответствующему типу операнда знакового типа

`unsigned int(4)+ long(4) -> unsigned long(4) + unsigned long(4)`

- ◆ Числа типа `float` не преобразуются автоматически к `double`

Старшинство операций

Операции	Ассоциативность
<code>! ++ -- + - sizeof (type)</code>	Справа налево
<code>* / %</code>	Слева направо
<code>+ -</code>	Слева направо
<code>== !=</code>	Слева направо
<code>&&</code>	Слева направо
<code> </code>	Слева направо
<code>= += -= *= /= %=</code>	Справа налево
<code>,</code>	Слева направо

Операторы

◆ **Выражение-оператор:** `expression;`

◆ **Составной оператор:** `{ }`

◆ **Условный оператор:** `if (expr) stmt; else stmt;`

◆ `else` всегда относится к ближайшему `if`:

```
if (x > 2)
    if (y > z)
        y = z;
    else
        z = y;

if (x > 2) {
    if (y > z)
        y = z;
}
else
    z = y;
```

◆ **Оператор выбора:** `switch (expr) {`
`case const-expr: stmt;`
`case const-expr: stmt;`
`default: stmt;`
`}`

◆ Оператор `break` – немедленный выход из `switch`.

Операторы

◇ Цикл *while*: `while (expression) stmt;`

◇ Цикл *for*:

```
for (decl1; expr2; expr3)      decl1;
    stmt;                      while (expr2) {
                                stmt;
                                expr3;
                                }
```

decl1 - ВОЗМОЖНО
определение переменной
с инициализатором

◆ `for (; ;) stmt;` – бесконечный цикл.

◇ Цикл *do-while*: `do { stmt; } while (expression);`

◆ Проверка условия выхода из цикла после выполнения тела.

◇ **Операторы *break* и *continue***: выход из внутреннего цикла и переход на следующую итерацию

◇ **Оператор *goto***: переход по метке
`goto label`

...

`label:`

◆ Областью видимости метки является вся функция

Пример программы. Количество дней между двумя датами

```
int main (void)
{
    while (1) {
        int m1, d1, y1, m2, d2, y2;
        int t1, t2;
        int days1, days2, total;

        if (scanf ("%d%d%d%d%d%d", &d1, &m1, &y1, &d2, &m2, &y2) != 6)
            break;
        t1 = check_date (d1, m1, y1);
        if (t1 == 1 || (t2 = check_date (d2, m2, y2)) == 1)
            break;
        else if (t1 == 2 || t2 == 2)
            continue;

        days1 = days_from_jan1 (d1, m1, y1);
        days2 = days_from_jan1 (d2, m2, y2);
        total = days_between_years (y1, y2) + (days2 - days1);
        printf ("Days between dates: %d, weeks between days: %d\n",
                total, total / 7);
    }
    return 0;
}
```

Пример программы. Количество дней между двумя датами

```
#include <stdio.h>

static int check_date (int d, int m, int y)
{
    if (!d || !m || !y)
        return 1;
    if (d < 0 || m < 0 || y < 0)
    {
        printf ("%d %d %d: wrong date\n", d, m, y);
        return 2;
    }
    return 0;
}

while (1) {
<...>
    t1 = check_date (d1, m1, y1);
    if (t1 == 1 || (t2 = check_date (d2, m2, y2)) == 1)
        break;
    else if (t1 == 2 || t2 == 2)
        continue;
<...>
}
```

Пример программы. Количество дней между двумя датами

```
static int leap_year (int y)
{
    return (y % 400 == 0) || (y % 4 == 0 && y % 100 != 0);
}
```

```
static int days_in_year (int y)
{
    return leap_year (y) ? 366 : 365;
}
```

```
static int days_between_years (int y1, int y2)
{
    int i;
    int days = 0;

    for (i = y1; i < y2; i++)
        days += days_in_year (i);
    return days;
}
```

Пример программы. Количество дней между двумя датами

```
static int days_from_jan1 (int d, int m, int y)
{
    int days = 0;

    switch (m) {
        case 12: days += 30;
        case 11: days += 31;
        case 10: days += 30;
        case 9: days += 31;
        case 8: days += 31;
        case 7: days += 30;
        case 6: days += 31;
        case 5: days += 30;
        case 4: days += 31;
        case 3: days += leap_year (y) ? 29 : 28;
        case 2: days += 31;
        case 1: break;
    }
    return days + d;
}
```