Курс «Алгоритмы и алгоритмические языки»

Лекции 6-7

Форматный ввод-вывод

```
#include <stdio.h>

int sum (int x, int y)
{
  return x + y;
}

int main (void)
{
  int s = 0;
  int a, b;

  scanf ("%d%d", &a, &b);
  s += sum (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;
    d = sqrt (d);
    printf ("Решение 1: %.4f, решение 2: %.4f\n", (db+d)/(2*a), (db-d)/(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;
    d = sqrt (d);
    printf ("Решение 1: %.4f, решение 2: %.4f\n",
            (db+d)/(2*a), (db-d)/(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
`! ++ -- + - sizeof (type)`	Справа налево
`* / %`	Слева направо
`+ -`	Слева направо
`== !=`	Слева направо
`&&`	Слева направо
`||`	Слева направо
`= += -= *= /= %=`	Справа налево
`,`	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 (x > 2) {
  if (y > z)                    if (y > z)
    y = z;                        y = z;
  else                          }
    z = y;                      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 (expr1; expr2; expr3)
  stmt;
```

```
expr1;
while (expr2) {
  stmt;
  expr3;
}
```

  - `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;
}
```

Символьный тип данных (char)

Программа подсчета числа строк во входном потоке

```
#include <stdio.h>
int main (void)
{
  int c, nl = 0;
  while ((c = getchar()) != EOF)
    if (c == ‘\n’)
      ++nl;
  printf ("%d\n", nl);
  return 0;
}
```

Символьный тип данных (char)

В стандарте языка Си символьные данные представляются в коде ***ASCII*** (**American Standard Code for Information Interchange**).

- Каждому символу сопоставляется его код – число типа **char**.
- Символы упорядочены в алфавитном порядке (для английского алфавита)
- К символьным данным применимы операции целочисленных типов (но обычно – ***операции отношения и сравнения***)
- Каждый символ, представляющий самого себя, заключается в одинарные кавычки ' и '
- Последовательность символов (строка) заключается в двойные кавычки " и "
- Специальные (управляющие) символы представляются последовательностями из двух символов. Примеры:
  - **`\n`** переход на начало новой строки
  - **`\t`** знак табуляции
  - **`\b`** возврат на один символ с затиранием

Символьный тип данных (char)

В стандарте языка Си символьные данные представляются в коде ***ASCII*** (**American Standard Code for Information Interchange**).

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0	\0									\t	\n						
1												ESC					
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL	

***Символьный тип данных* (char)**

В стандарте языка Си символьные данные представляются в коде ***ASCII*** (**American Standard Code for Information Interchange**).

- В коде *ASCII* буквы верхнего и нижнего регистра составляют непрерывные последовательности:

  между **a** и **z** (соответственно, между **A** и **Z**) нет ничего, кроме букв, расположенных в алфавитном порядке.
- Это же верно и для цифр **0**, **1**, ..., **9**
- Преобразование строки символов цифр **s** в целое число

```
int atoi (char s[])
{
  int i, n;
  n = 0;
  for (i = 0; s[i] >= '0' && s[i] <= '9'; ++i)
      n = 10 * n + (s[i] – '0');
   return n;
}
```

Массивы

- Массивы позволяют организовывать непрерывные последовательности нескольких однотипных элементов и обращаться к ним по номеру (индексу).
- Элементы массивов располагаются в памяти последовательно и индексируются с 0:

```
int a[30];  /* элементы a[0], a[1], … , a[29] */
```

- Все массивы – одномерные, но элементом массива может быть массив:

```
int b[3][3]; /* элементы b[0][0], b[0][1], b[0][2],
                          b[1][0], b[1][1], b[1][2],
                          b[2][0], b[2][1], b[2][2]
                */
```

- ***Пример***. Программа, подсчитывающая количество вхождений в строку (текст) каждой из десяти цифр (**`ndigit[10]`**), пробельных символов (**`nwhite`**) и остальных символов (**`nother`**).

Массивы

```
#include <stdio.h>
int main (void)
{
  int c, i, nwhite, nother, ndigit[10];
  nwhite = nother = 0;
  for (i = 0; i < 10; ++i)
    ndigit[i] = 0;
  while (c = getchar())!= EOF)
    if (c >= ‘0’ && c <= ‘9’)
      ++ndigit[c – ‘0’];
    else if (c == ‘ ’ || c == ‘\n’ || c == ‘\t’)
      ++nwhite;
    else
      ++nother;
  printf("digits=");
  for (i = 0; i < 10; ++i)
    printf(" %d", ndigit[i]);
  printf(", white space=%d, other=%d\n”, nwhite, nother);
  return 0;
}
```

Инициализация массивов

***тип имя_массива*[*размер1*]…[*размерN*] = {*список_значений*};**

Пример (справа для наглядности использованы дополнительные фигурные скобки – ***группировка подагрегатов***)

```
int sqrs[10][2] = {
        1, 1,
        2, 4,
        3, 9,
        4, 16,
        5, 25,
        6, 36,
        7, 49,
        8, 64,
        9, 81,
        10, 100
}
```

```
int sqrs[10][2] = {
        {1, 1},
        {2, 4},
        {3, 9},
        {4, 16},
        {5, 25},
        {6, 36},
        {7, 49},
        {8, 64},
        {9, 81},
        {10, 100}
}
```

Строки

- ***Строка*** – это одномерный массив типа **`char`**

  Объявляя массив, предназначенный для хранения строки, необходимо предусмотреть место для символа **`'\0'`** (конец строки)

- ***Строковая константа*** (например, **`"string"`**).
  В конец строковой константы компилятор добавляет **`'\0'`**.

- Стандартная библиотека функций работы со строками **`<string.h>`**, в частности, содержит такие функции, как:
  - **`strcpy(s1, s2)`** (копирование **`s2`** в **`s1`**)
  - **`strcat(s1, s2)`** (конкатенация **`s2`** и **`s1`**)
  - **`strlen(s)`** (длина строки **`s`**)
  - **`strcmp(s1, s2)`** (сравнение **`s2`** и **`s1`** в *лексикографическом порядке*: 0, если **`s1`**и **`s2`** совпадают, отрицательное значение, если **`s1 < s2`**, положительное значение, если **`s1 > s2`**)
  - **`strchr(s, ch)`** (указатель на первое вхождение символа **`ch`** в **`s`**)
  - **`strstr(s1, s2)`** (указатель на первое вхождение подстроки **`s2`** в строку **`s1`**)

Строки

```
#include <stdio.h>
#include <string.h>
int main (void)
{
  char string1[80], string2[80], smp[3];
  fgets (string1, 80, stdin); string1[strlen (string1)-1] = '\0';
  fgets (string2, 80, stdin); string2[strlen (string2)-1] = '\0';

  printf("Строки имеют длину: первая %d, вторая %d\n,
         strlen (string1), strlen (string2));
  if(!strcmp (string1, string2))
    printf ("строки равны\n");
  strncat (string1, string2, 80 – strlen (string1) - 1);
  printf ("%s\n", string1);
  strcpy (string1, "Привет, ");
  printf ("%s\n", string1);
  if (strchr (string1, 'и'))
    printf("Буква 'и' есть в %s\n", string1);
  smp[0] = 'и'; smp[1] = 'в'; smp[2] = 0;
  if (strstr (string1, smp))
    printf("Есть %s\n", smp);
  return 0;
}
```

Строки

```
#include <stdio.h>
#include <string.h>
int main (void)
{
  char string1[80], string2[80], smp[3];
  fgets (string1, 80, stdin); string1[strlen (string1)-1] = '\0';
  fgets (string2, 80, stdin); string2[strlen( string2)-1] = '\0';

  printf("Строки имеют длину: первая %d, вторая %d\n,
           strlen (string1), strlen (string2));
  if(!strcmp (string1, string2))
    printf ("строки равны\n");
  strncat (string1, string2, 80 – strlen (string1) - 1);
  printf ("%s\n", string1);
  strcpy (string1, "Привет, ");
  printf ("%s\n", string1);
  if (strchr (string1, 'и'))
    printf("Буква 'и' есть в %s\n", string1);
  smp[0] = 'и'; smp[1] = 'в'; smp[2] = 0;
  if (strstr (string1, smp))
    printf("Есть %s\n", smp);
  return 0;
}
```

Если **string1** – **"Здравствуй, "** , а **string2** –**"мир!"**, результат:

Строки имеют длину 12 4

Здравствуй, мир!

Привет,

Буква и есть в Привет

Есть ив