

**Курс «Алгоритмы и алгоритмические языки»
1 семестр 2018/2019**

Лекция 8

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;
}
```

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;
}
```

Каков должен быть возвращаемый тип функции `getchar`?

Символьный тип данных (char)

Символьные данные представляются в некотором коде. Популярным кодом является ASCII (American Standard Code for Information Interchange).

- ◆ Каждому символу сопоставляется его код – число типа `char`
- ◆ **Требуется**, чтобы в кодировке присутствовали маленькие и большие английские буквы, цифры, некоторые другие символы
- ◆ **Требуется**, чтобы коды цифр 0, 1, ..., 9 были последовательны
- ◆ К символьным данным применимы операции целочисленных типов (но обычно – **операции отношения и сравнения**)
- ◆ Каждый символ-литерал 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)

Таблица ASCII

	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 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 (верно для любой кодировки символов)

```
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;  
}
```