

Семинар №3

ФАКИ 2015

Бирюков В. А.

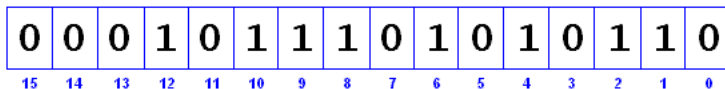
October 14, 2016

Переменные

Переменные

- В языке C все переменные нужно объявить перед использованием
- При объявлении – выделяется память под переменную
- Области видимости переменной
- Название переменной может содержать латинские буквы, цифры и _
- Название переменной не может начинаться с цифры

Целочисленные типы



Число бит на тип зависит от компилятора. Обычные значения такие:

Название типа	Число бит	Макс. значения
char	8	0..255
short	16	-32768..32767
int	32	$-2 \cdot 10^9$.. $+2 \cdot 10^9$
long	32	$-2 \cdot 10^9$.. $+2 \cdot 10^9$
long long	64	-2^{64} .. $+2^{64} - 1$

Беззнаковые целочисленные типы

Число бит на тип зависит от компилятора. Обычные значения такие:

Название типа	Число бит	Макс. значения
unsigned short	16	0..65535
unsigned int	32	0 .. $+4 \cdot 10^9$
unsigned long	32	0 .. $+4 \cdot 10^9$
unsigned long long	64	0 .. $+2^{65} - 1$

sizeof() – размер файла в байтах

Типы чисел с плавающей точкой



Название типа	Число бит	Макс. значения
float	32	$10^{-38}..10^{+38}$
double	64	$10^{-308}..10^{+308}$

Обычно используется double, так как float может недостаточно точен

Вывод в stdout. Функция printf.

printf(строка форматирования, пер1, пер2, ...)

Обозначение	Типы	Пример
d или i	Целочисленные типы	392
f	Типы с плавающей точкой	392.5
e	Научная нотация	3.9265e+2
c	Символ	a
s	Строка	HelloMipt!

Приоритет операторов

- 1 `()`, `[]`
- 2 `++`, `--`, `+`, `-` (унарные), `sizeof`
- 3 `*`, `/`, `%`
- 4 `+`, `-`
- 5 `>`, `<`, `<=`, `>=`
- 6 `==`, `!=`
- 7 `&`, `|`, `&&`, `||`
- 8 `=`, `+=`, и т.д.

Приоритет операторов C подробнее:

ru.cppreference.com/w/c/language/operator_precedence

Управляющие конструкции

Базовые управляющие конструкции

Цикл while

```
int i = 0;

while (i < 3)
{
    i++;
    printf("%d ", i);
}
```

Напечатает 1 2 3

Базовые управляющие конструкции

Цикл do while

```
int i = 0;

do
{
    i++;
    printf( "%d ", i );
} while ( i < 3 );
```

Напечатает 1 2 3

Базовые управляющие конструкции

Цикл for

```
for (int i = 0; i < 3; ++i) {  
    printf("%d ", i);  
}
```

Напечатает 1 2 3

Управляющие конструкции

Оператор break

```
for (int i = 0; i < 10; ++i) {  
    if (i == 6) {  
        break;  
    }  
    printf ("%d ", i);  
}
```

Управляющие конструкции

Оператор continue

```
for (int i = 0; i < 10; ++i) {  
    if (i == 6) {  
        continue;  
    }  
    printf ("%d ", i);  
}
```

Массивы и строки

Массивы

Примеры

Объявление:

```
int array [10];  
float average_temperature [12];
```

Доступ к элементу:

Нумерация в массиве начинается с 0

```
printf ("%d\n", array [9]);  
average_temperature [2] = 5.2;
```


Массивы

Инициализация

```
int array[10];  
for (int i = 0; i < 10; ++i) {  
    array[i] = /* something */;  
}
```

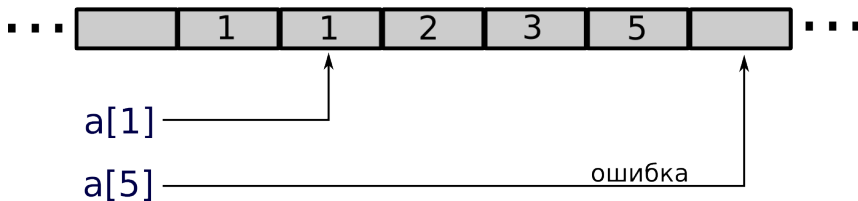
Или так:

```
int array[5] = {1, 1, 2, 3, 5};
```

Массивы

Массивы в памяти

```
int a[5] = {1, 1, 2, 3, 5};
```



ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Строки

Инициализация

```
int string[10];  
for (int i = 0; i < 10; ++i) {  
    string[i] = /* something */;  
}
```

Или так:

```
int string[10] = { 'h', 'e', 'l', 'l',  
    'o', '\0', 'a', 'b', 'c', 'd' };
```

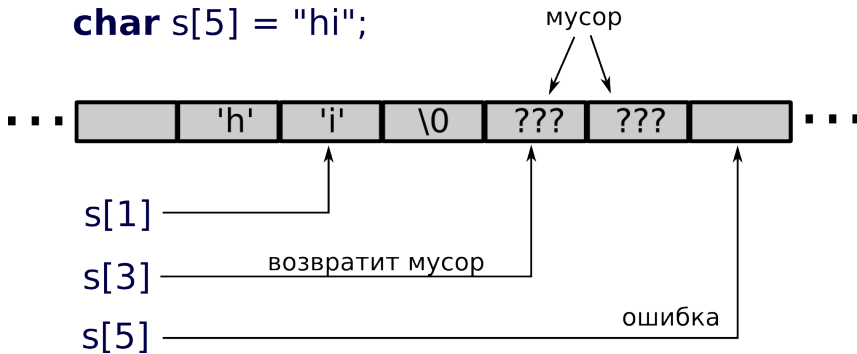
Или так:

```
int string[10] = "hello";
```

Строки

Строки в памяти

```
char s[5] = "hi";
```



Функции для работы со строками

Чтение строки:

```
char string[10];  
scanf("%10s", string);
```

Длина строки:

```
#include <string.h>  
char string[10] = "hello";  
int n = strlen(string);
```

Функции для работы со строками

```
#include <string.h>
char s1[10] = "hi";
char s2[10] = "world";
```

Копирование строки s2 в строку s1:

```
strcpy(s1, s2);
```

Конкатенация строки s2 в строку s1:

```
strcat(s1, s2);
```

Функций

Объявление функций(прототипы функций)

- Как и переменная, функция также должна объявлена перед использованием
- Определение функции называется прототипом функции
- Пример прототипа:

```
int sum (int a, int b)
```

или:

```
int sum (int , int )
```

Вызов функций

- Примеры вызова функции:

Функция, определяемая пользователем:

```
sum(a, b)
```

Библиотечные функции:

```
sqrt(x)
```

```
printf("%d\n", a)
```

Определение функции

Возвращаемый
тип

Имя функции

Параметры
функции

```
int sum ( int a, int b )
```

```
{
```

```
    int c;
```

Объявление локальной
переменной

```
    c = a + b;
```

```
    return ( c );
```

```
}
```

Оператор return
завершает выполнение функции
возвращает значение c

Функции

int sum(**int** a, **int** b); ← Прототип функции

```
int main()
{
    printf("%d\n", sum(5, 4));
    return 0;
}
```

↑ Вызов функции


```
int sum(int a, int b)
{
    return a + b;
}
```

← Определение функции

Функции

```
int sum(int a, int b)
{
    return a + b;
}
```

Прототип и
определение
функции



```
int main()
{
    printf("%d\n", sum(5, 4));
    return 0;
}
```

Вызов функции



Области видимости переменных

Области видимости переменных

- Область видимости – область программы, в пределах которой имя некоторой переменной продолжает быть связанным с этой переменной и возвращать её значение.
- Глобальная переменная – объявляются вне всех функций и доступны отовсюду
- Локальная переменная – объявляются внутри блока и недоступны вне его

Области видимости переменных

Функция определяет собственную (локальную) область видимости, куда входят:

- 1 Глобальные переменные
- 2 Входные параметры
- 3 Переменные, которые объявляются в теле самой функции

Материалы для подготовки

- Керниган, Ричи. Язык C
- CS50 (перевод лекций есть в vk)
- <http://style.vdi.mipt.ru/> – тренировка перед контрольной работой