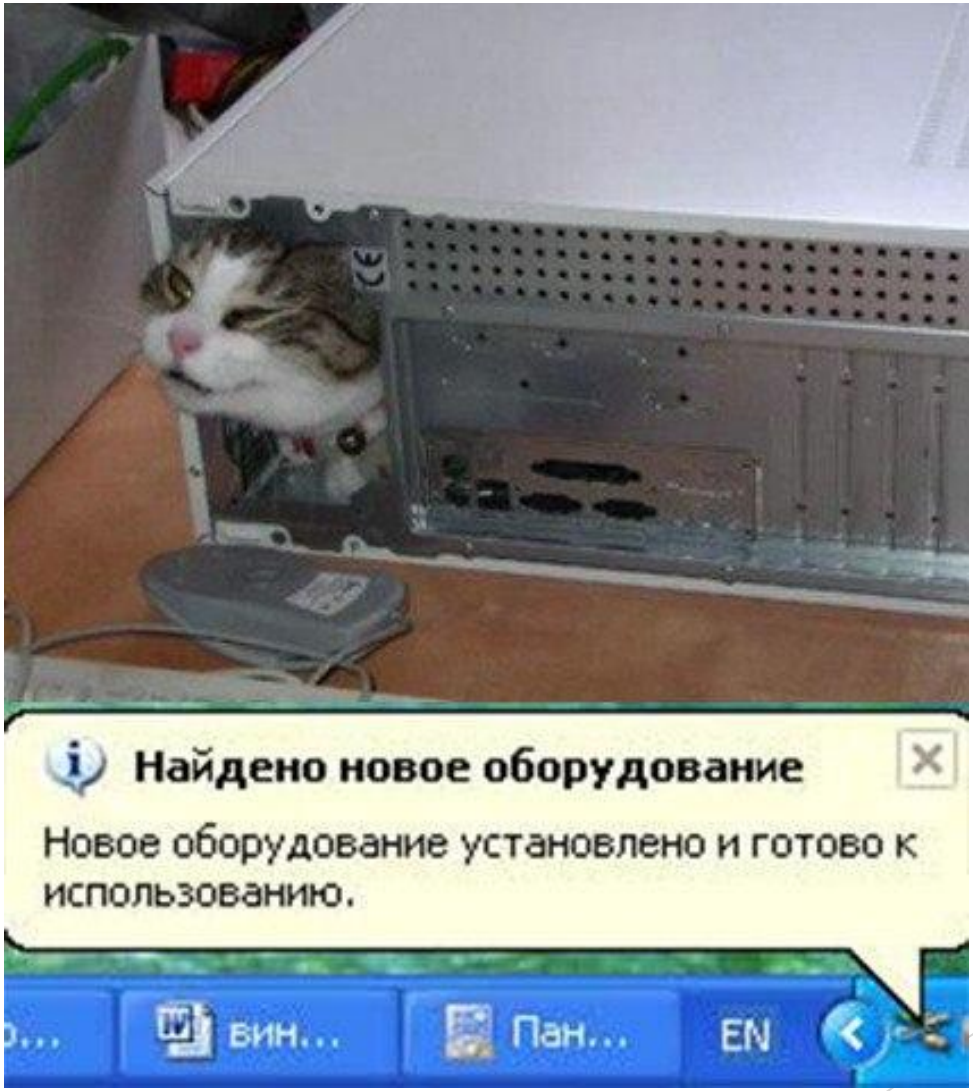
Основы программирования на C++

Лекция 4. Строки

Повторение.

- Что такое указатель? Как его объявить?
- Как указать на переменную? Пример
- Арифметика указателей
- Что такое ссылка?
- Как выделяется память в C++? Как в C?
- Как создать динамический массив? Как создать N-мерный динамический массив? Пример.
- Задача: Дан целочисленный массив [2, 5, 3, 6, 7, 9, 0]. Напишите функцию, возвращающую указатель на цифру 7 в массиве. Через полученный указатель выведите все дальнейшие цифры. Массив кончается нулём.

Мем в начале


Найдено новое оборудование
Новое оборудование установлено и готово к использованию.
вин...
Пан...
EN

Символы и строки. Как?

Строка — последовательность (массив) символов. Если в выражении встречается одиночный символ, он должен быть заключен в одинарные кавычки. При использовании в выражениях строка заключается в двойные кавычки. Признаком конца строки является нулевой символ **\0**.

Объявляться строка может так:

- `char *string = "Hello, guys!";`
- `char str[80] = "Hello, guys!";`
- `char s[] = "Hello, guys!";`
- `char *st = new char[80];`

  `strcpy(st, "Hello, guys!“);`
- `char mas[12] = { 'H', 'e', 'l', 'l', 'o', ' ', 'g', 'u', 'y', 's' , '!','\0' };`

Таблица ASCII

Стандартная часть кода ASCII

32		00100000	56	8	00111000	80	P	01010000	104	h	01101000
33	!	00100001	57	9	00111001	81	Q	01010001	105	i	01101001
34	"	00100010	58	:	00111010	82	R	01010010	106	j	01101010
35	#	00100011	59	;	00111011	83	S	01010011	107	k	01101011
36	$	00100100	60	<	00111100	84	T	01010100	108	l	01101100
37	%	00100101	61	=	00111101	85	U	01010101	109	m	01101101
38	&	00100110	62	>	00111110	86	V	01010110	110	n	01101110
39	'	00100111	63	?	00111111	87	W	01010111	111	o	01101111
40	(	00101000	64	@	01000000	88	X	01011000	112	p	01110000
41	)	00101001	65	A	01000001	89	Y	01011001	113	q	01110001
42	*	00101010	66	B	01000010	90	Z	01011010	114	r	01110010
43	+	00101011	67	C	01000011	91	[	01011011	115	s	01110011
44	,	00101100	68	D	01000100	92	\	01011100	116	t	01110100
45	-	00101101	69	E	01000101	93	]	01011101	117	u	01110101
46	.	00101110	70	F	01000110	94	^	01011110	118	v	01110110
47	/	00101111	71	G	01000111	95	_	01011111	119	w	01110111
48	0	00110000	72	H	01001000	96	`	01100000	120	x	01111000
49	1	00110001	73	I	01001001	97	a	01100001	121	y	01111001
50	2	00110010	74	J	01001010	98	b	01100010	122	z	01111010
51	3	00110011	75	K	01001011	99	c	01100011	123	{	01111011
52	4	00110100	76	L	01001100	100	d	01100100	124	|	01111100
53	5	00110101	77	M	01001101	101	e	01100101	125	}	01111101
54	6	00110110	78	N	01001110	102	f	01100110	126	~	01111110
55	7	00110111	79	O	01001111	103	g	01100111	127	□	01111111

Необходимо понимать

Символьной переменной можно присваивать код символа из таблицы ASCII. Также можно из кода символа или его самого отнимать число, чтобы поменять его код. Например:

```
char z = 'z';
char a = z - 25;
cout << a << endl;
```

Как с ними работать?

Для работы со строками существует специальная библиотека **string.h** (или **cstring**). НЕ ПУТАТЬ С БИБЛИОТЕКОЙ **string.**

Ссылки на описание библиотеки **string.h**:

- https://ru.wikipedia.org/wiki/String.h
- http://www.cplusplus.com/reference/cstring/
- http://cppstudio.com/cat/309/325/

Функция strlen

Длина С-строки определяется по достижению нулевого символа **‘\0’** – нуль терминатор. Функция **strlen** видит начало Си-строки и начинает сначала считать количество символов (байтов, отводимых под каждый символ), этот процесс выполняется до тех пор, пока не будет достигнут завершающий нулевой символ.

Прототип функции:

```
size_t strlen(const char * string);
```

Пример работы:

```
char *string = "1234567890";
cout << strlen(string) << endl;
```

Функция strcpy

```
char * strcpy(char * destptr, const char * srcptr);
```

```
char * strncpy(char * destptr, const char * srcptr, size_t num);
```

Функция копирует строку **srcptr**, включая завершающий нулевой символ в строку назначения, на которую ссылается указатель **destptr**.

Пример:

```
char *string = "This is string";
char *s=new char[15];
strcpy(s, string);
cout << s << endl;
```

Функция strcat

```
char * strcat(char * destptr, const char * srcptr);
```

Соединяет обе строки в одну **destptr**, добавляя строку **srcptr** в конец **destptr**.

```
char *s1= "This ";
char *s2 = "is ";
char *s3 = "string";
char *s=new char[15];
s = "";
strcat(s, s1);
strcat(s, s2);
strcat(s, s3);
cout << s << endl;
```

Функция strcmp

```
int strcmp(const char * string1, const char
* string2);
```

```
int strncmp(const char * string1, const char
* string2, size_t num);
```

Функция сравнивает строки и возвращает разницу в строках. Начиная с первых символов функция сравнивает поочередно каждую пару символов, и продолжается это до тех пор, пока не будут найдены различные символы или не будет достигнут конец строки.

Функция возвращает несколько значений, которые указывают на отношение строк:

- Нулевое значение говорит о том, что обе строки равны.
- Значение больше нуля указывает на то, что строка string1 больше строки string2, значение меньше нуля свидетельствует об обратном.

Пример - проверка пароля

```
char *password = "123";
cout << "Enter the password: ";
char input[30];
cin >> input;
if (strcmp(input, password) == 0)
cout << "Correct" << endl;
else cout << "Incorrect password" <<
endl;
```

Функция strstr

```
char* strstr(const char *string1, const char *string2);
```

Функция возвращает указатель на первое вхождение строки string2 в строку string1.

Нулевой указатель, если последовательность символов строки string2 не входит в string1.

```
char *string1 = "123";
char *string2 = "Тут где-то находится числа 123 & 456";
cout << strstr(string2, string1) << endl;
```

Функции atoi, atof, atol

Данные функции преобразуют строковую переменную в целочисленные и вещественные типы данных:

- `atoi(const char *string);` - преобразует строку к типу `int`
- `atof(const char *string);` - преобразует строку к типу `float`
- `atol(const char *string);` - преобразует строку к типу `long`

Все данные функции возвращают типы, к которым они преобразуют строку.

Обратная функция - itoa_s

```
char *_itoa_s(int value, char * string, int size,
int radix);
```

Данная функция принимает значение и выходную строку, размер строки и основание системы счисления.

```
int number = 228;

int radix = 16;

char strToNum[5];

_itoa_s(number, strToNum,
sizeof(strToNum), radix);

cout << strToNum << endl;
```

В итоге, что должны знать Краткий конспект

- strlen – длина строки
- strcat – соединение строк
- strcpy – копирование строки
- strcmp – сравнение строк
- strstr – поиск подстроки в строке
- atoi – перевод строки в число
- _itoa_s – перевод числа в строку

Задачи

- Функция 1. Введите строку и сообщите, сколько слов в ней. Считайте, что все слова в строке разделены ровно одним пробелом.
- Функция 2. Напишите программу, которая принимает строку и проверяет, является ли она палиндромом.
- Функция 3. Написать программу, которая проверяет, является ли введенная с клавиатуры строка шестнадцатеричным числом.
- Функция 4. Напишите программу, которая вычисляет введенное пользователем значение выражения типа $A_1\ Z_1\ A_2\ Z_2\ A_3\ Z_3 \ldots A_i\ Z_j$, где A_i — целое число от 0 до 9, а Z_j — знак + или -.

Структуры

Структура - это совокупность переменных, объединенных одним именем, предоставляющая общепринятый способ совместного хранения информации. Объявление структуры приводит к образованию шаблона, используемого для создания объектов структуры. Переменные, образующие структуру, называются членами структуры. (Члены структуры также часто называются элементами или полями.)

Иначе говоря, структура – это определяемый программистом тип данных, состоящий из основных типов и уже существующих определенных типов.

Конструкция

```
struct Name
{
    type atrib;
    // остальные элементы структуры
} structVar1, structVar2, ...;
```

Где:

- `Name` – название структуры
- `type atrib;` - поле данных. Их может быть несколько
- `structVar1, structVar2` – глобальные переменные типа этой структуры

Пример

```
struct student
{
    char name[20];
    char studak[15];
    char facultet[6];
    int age;
    double stipendia;
};
int main() {
    student Vasya;
    int size = sizeof(Vasya);
    cout << size << endl;
    system("pause");
}
```

Обращение к полям

Если объект структуры создан обычной переменной, то обращение к полю происходит как:

имя_переменной.имя_поля;

Если объект структуры создан через указатель, то обращение к полю происходит как:

имя_переменной->имя_поля;

Пример:

```
Vasya.age = 18;
Vasya.stipendia = 1337.228;
strcpy(Vasya.name, "Vasya Pupkin");
student *Petya = new student();
Petya->age = 17;
(*Petya).stipendia = 144.8;
```

Также разрешено инициализировать структуры подобным образом:

```
student Lena = { "Lena", "01013789", "AVTF", 16, 1337.4 };
```

Структуры могут быть аргументом функции и ещё возвращаемым значением.

```
void show(student &obj)
{
cout << "Name:       " << obj.name << endl;
cout << "Studak:     " << obj.studak << endl;
cout << "Facultet:   " << obj.facultet << endl;
cout << "Age:        " << obj.age << endl;
cout << "stipendia:  " << obj.stipendia << endl;
}
```

Задачки на разбор

Дан тип комплексного числа:

```
struct Complex
{
    int a;
    int b;
};
```

Написать функции:

- Показа числа в правильном виде
- Сложение
- Вычитание
- Деление
- Сравнение двух комплексных чисел

Ещё задача на разбор

Создайте структуру окружности через структуру точки и для неё функции:

- длины окружности
- площади окружности
- площади сектора с углом A.

Мем в конце

How people run	

How programmer run	public void run()
	{
	step++;
	}