

The background features abstract, overlapping geometric shapes in various shades of green, ranging from light lime to dark forest green. These shapes are primarily located on the left and right sides of the frame, leaving a large white central area. The shapes are composed of triangles and polygons, some of which are semi-transparent, creating a layered effect.

Строки

Кодировка ASCII

Все символы представляются своим кодом. То, какой код какому символу соответствует, задает кодировка. В качестве основной, обычно используется ASCII (“American Standard Code for Information Interchange”). Один символ в ASCII занимает 1 байт. Первые 32 символа в ASCII - служебные, вроде нулевого (NUL) или перевода строки (LF) .

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
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

Строки

Строки представляют собой массив (последовательность) символов, причем, чаще всего используются нуль-терминированные строки (за последним символом строки идет специальный символ - '\0'). Строка задается двойными кавычками.

Работа со строками обеспечивается стандартной библиотекой `string`, которую можно подключить посредством `#include <string>`.

```
1 #include <string>
2
3 std::string example;
```

```
1 #include <string>
2 using namespace std;
3
4 string example;
```

Операции над строками

- ▶ Присваивание
- ▶ Сравнение
- ▶ Получение i -го символа
- ▶ Ввод
- ▶ Запрос длины
- ▶ Конкатенация
- ▶ Удаление/вставка/копирование фрагмента
- ▶ Поиск подстроки
- ▶ Преобразования к числу/строке

Присваивание (через =)

```
1 string example = "Do you speak English?"; // присваивание при создании
2
3 example = ""; // присваивание пустой строки,
4 // теперь example хранит другие данные - ""
5
6 string test = example; // присваивание между строками
7 // данные из example будут скопированы в test
```

Ввод

Для ввода строк можно использовать функцию `getline()`. Данная функция читает строку целиком (с пробелами и прочими символами) до знака перевода строки.

Также строку можно вводить через `cin` (до первого разделительного символа).

```
1 string example;
2
3 getline(cin, example);
4 // либо
5 cin >> example;
```

Сравнение

Две строки равны друг другу, если у них одинаковые длины и символы на соответствующих местах полностью совпадают.

Выяснение какая из строк больше выполняется посимвольно. Если строка начинается с символа, чей код больше, то и сама строка больше. Если первые символы одинаковы, то сравниваются вторые символы по тому же правилу. Если и они одинаковы, сравниваются третьи и так далее. Если в процессе сравнения какая-то из строк закончилась, то она меньше другой.

```
1 string a = "Hello", b = "World";  
2  
3 bool x = (a < b); // true  
4 bool y = (a > b); // false  
5 bool z = (a == b); // false  
6 bool w = (a != b); // true
```

Запрос длины

Ничего сложного - возвращает длину строки. Метод `size()`.

```
1 string example = "Hello!";  
2 int x = example.size(); // x, очевидно, равен 6
```

Конкатенация

«Склеивает» строки. Так "a" + "b" = "ab".

```
1 string example1 = "Do you ";  
2 string example2 = "speak ";  
3 string example3 = "French?";  
4  
5 cout << example1 + example2 + example3 << endl;  
6 // Do you speak French?
```

Получение i-го символа

Посредством `[i]`, возвращает i-ый символ строки.

```
1 string example = "Nice string";  
2 char x = example[5]; // x = 's'
```


Вставка фрагмента

Выполняется методом `insert(позиция, фрагмент)`.

```
1 example = "123456";
2 example.insert(3, " + ");
3 // теперь example str = "123 + 456"
```

Удаление фрагмента и копирование

Удаление выполняется методом `erase(позиция, количество символов)`.

Копирование - методом `substr(позиция, количество символов)`.

```
1 string example = "Hello, world!";
2 string new_example = example.substr(3, 6);
3 example.erase(3, 6);
4
5 // new_example = "lo, wo"
6 // example = "Helrld!"
```


Поиск подстроки

С помощью метода `find()` можно определить, встречается ли данный фрагмент в строке. Функция возвращает специальное значение `string::npos`, если фрагмент не найден или номер первого (начиная слева) места, откуда начинается подобный фрагмент.

```
1 example = "Hello!";
2 x = example.find("z");
3 y = example.find("H");
4 // Здесь x = npos, а y = 0
```

Преобразования числа к строке или строки к числу

Число к строке преобразуется функцией `to_string(число)`.

Строка к числу преобразуется функцией `stoi(строка)`.

```
1 double f = 23.43;
2 string f_str = to_string(f);
3
4 example str = "12";
5 int x = stoi(str);
```