Программирование на языке C++

...

Строки языка Си

Строки в языке Си

Строка - это последовательность ASCII или UNICODE символов

```
#include <cstring>
#include <iostream>

int main()
{
   char cstr[25] = "I'm Plain C string!"; // 19 symbols

   std::cout << cstr << std::endl;
   std::cout << "My size is " << strlen(cstr) << "\n";
   std::cout << "But my array size is " << sizeof(cstr) << "\n";
   std::cout << "One symbol size is " << sizeof(cstr[0]) << "\n";
}
```

Таблица ASCII

American Standard Code for Information Interchange

Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char	Dec	Bin	Hex	Char
0	0000 0000	00	[NUL]	32	0010 0000	20	space	64	0100 0000	40	@	96	0110 0000	60	`
1	0000 0001	01	[SOH]	33	0010 0001	21	!	65	0100 0001	41	A	97	0110 0001	61	a
2	0000 0010	02	[STX]	34	0010 0010	22	"	66	0100 0010	42	B	98	0110 0010	62	b
3	0000 0011	03	[ETX]	35	0010 0011	23	#	67	0100 0011	43	C	99	0110 0011	63	c
4	0000 0100	04	[EOT]	36	0010 0100	24	$	68	0100 0100	44	D	100	0110 0100	64	d
5	0000 0101	05	[ENQ]	37	0010 0101	25	%	69	0100 0101	45	E	101	0110 0101	65	e
6	0000 0110	06	[ACK]	38	0010 0110	26	&	70	0100 0110	46	F	102	0110 0110	66	f
7	0000 0111	07	[BEL]	39	0010 0111	27	'	71	0100 0111	47	G	103	0110 0111	67	g
8	0000 1000	08	[BS]	40	0010 1000	28	(	72	0100 1000	48	H	104	0110 1000	68	h
9	0000 1001	09	[TAB]	41	0010 1001	29	)	73	0100 1001	49	I	105	0110 1001	69	i
10	0000 1010	0A	[LF]	42	0010 1010	2A	*	74	0100 1010	4A	J	106	0110 1010	6A	j
11	0000 1011	0B	[VT]	43	0010 1011	2B	+	75	0100 1011	4B	K	107	0110 1011	6B	k
12	0000 1100	0C	[FF]	44	0010 1100	2C	,	76	0100 1100	4C	L	108	0110 1100	6C	l
13	0000 1101	0D	[CR]	45	0010 1101	2D	-	77	0100 1101	4D	M	109	0110 1101	6D	m
14	0000 1110	0E	[SO]	46	0010 1110	2E	.	78	0100 1110	4E	N	110	0110 1110	6E	n
15	0000 1111	0F	[SI]	47	0010 1111	2F	/	79	0100 1111	4F	O	111	0110 1111	6F	o
16	0001 0000	10	[DLE]	48	0011 0000	30	0	80	0101 0000	50	P	112	0111 0000	70	p
17	0001 0001	11	[DC1]	49	0011 0001	31	1	81	0101 0001	51	Q	113	0111 0001	71	q
18	0001 0010	12	[DC2]	50	0011 0010	32	2	82	0101 0010	52	R	114	0111 0010	72	r
19	0001 0011	13	[DC3]	51	0011 0011	33	3	83	0101 0011	53	S	115	0111 0011	73	s
20	0001 0100	14	[DC4]	52	0011 0100	34	4	84	0101 0100	54	T	116	0111 0100	74	t
21	0001 0101	15	[NAK]	53	0011 0101	35	5	85	0101 0101	55	U	117	0111 0101	75	u
22	0001 0110	16	[SYN]	54	0011 0110	36	6	86	0101 0110	56	V	118	0111 0110	76	v
23	0001 0111	17	[ETB]	55	0011 0111	37	7	87	0101 0111	57	W	119	0111 0111	77	w
24	0001 1000	18	[CAN]	56	0011 1000	38	8	88	0101 1000	58	X	120	0111 1000	78	x
25	0001 1001	19	[EM]	57	0011 1001	39	9	89	0101 1001	59	Y	121	0111 1001	79	y
26	0001 1010	1A	[SUB]	58	0011 1010	3A	:	90	0101 1010	5A	Z	122	0111 1010	7A	z
27	0001 1011	1B	[ESC]	59	0011 1011	3B	;	91	0101 1011	5B	[	123	0111 1011	7B	{
28	0001 1100	1C	[FS]	60	0011 1100	3C	<	92	0101 1100	5C	\	124	0111 1100	7C	|
29	0001 1101	1D	[GS]	61	0011 1101	3D	=	93	0101 1101	5D	]	125	0111 1101	7D	}
30	0001 1110	1E	[RS]	62	0011 1110	3E	>	94	0101 1110	5E	^	126	0111 1110	7E	~
31	0001 1111	1F	[US]	63	0011 1111	3F	?	95	0101 1111	5F	_	127	0111 1111	7F	[DEL]

Строки в языке Си

Строковый тип данных в C как таковой отсутствует, а в качестве строк в C используются обычные **массивы символов**.

```
#include <cstring>
#include <iostream>

int main()
{
   char cstr[25] = "I'm Plain C string!"; // 19 symbols

   std::cout << cstr << std::endl;
   std::cout << "My size is " << strlen(cstr) << "\n";
   std::cout << "But my array size is " << sizeof(cstr) << "\n";
   std::cout << "One symbol size is " << sizeof(cstr[0]) << "\n";
}
```

```
> ./CString.exe

I'm Plain C string!
My size is 19
But my array size is 25
One symbol size is 1
```

Размещение строки

```
char cstr[25] = "I'm Plain C string!"; // 19 symbols

strlen(cstr); // 19

sizeof(cstr); // 25
```

Размер строки и терминальный ноль

Длина строки указывается с учетом одного символа на хранение завершающего нуля, поэтому максимальное количество значащих символов в строке на единицу меньше ее длины

```
char cstr[] = "I'm Plain C string!"; // 19 symbols

strlen(cstr); // 19

sizeof(cstr); // 20
```

Строки в языке Си

Строки на языке Си являются **массивами символов**, поэтому к любому символу можно обратиться по его **индексу**.
Для этого используется синтаксис обращения к элементу массива

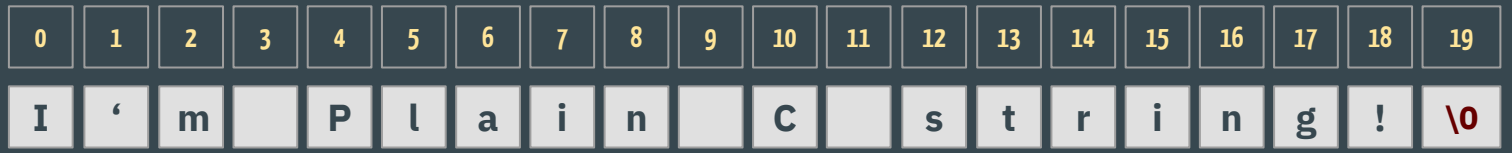
```
int main()
{
    char cstr[] = "I'm Plain C string!"; // 19 symbols
    cstr[7] = 'y';
    cstr[8] = 'i';
    cstr[9] = 'n';
    cstr[10] = 'g';
    cstr[11] = '\0';

    std::cout << cstr << std::endl;
    std::cout << "My size is " << strlen(cstr) << "\n";
    std::cout << "But my array size is " << sizeof(cstr) << "\n";
}
```

```
> ./CString2.exe

I'm Playing
My size is 11
But my array size is 20
```

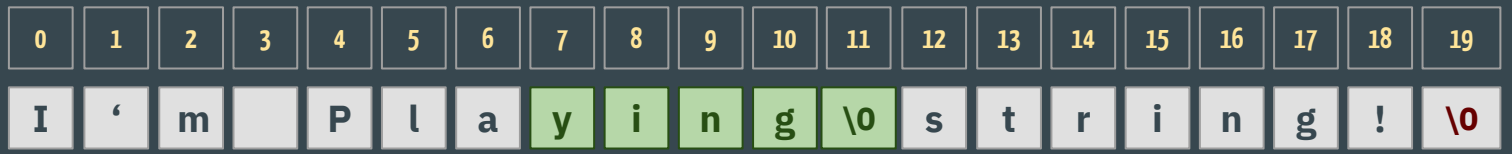
Размещение строки в памяти

Размер массива, выделенного под Cu-строку после изменения размера самой строки остается неизменным

```
char cstr[] = "I'm Plain C string!"; // 19 symbols
```




```
cstr[7] = 'y'; cstr[8] = 'i'; cstr[9] = 'n'; cstr[10] = 'g'; cstr[11] = '\0';
```



Строки в языке Си

Размер массива для произвольной строки

```
#include <cstring>
#include <iostream>

int main()
{
   char carBrand[10] = "";
   std::cout << "Please, enter your favorite car brand: ";
   std::cin >> carBrand;

   std::cout << "Array  size is " << sizeof(carBrand) << "\n";
   std::cout << "String size is " << strlen(carBrand) << "\n";
}
```

```cpp
#include <cstring>
#include <iostream>

int main()
{
   char carBrand[10] = "";
   std::cout << "Please, enter your favorite car brand: ";
   std::cin >> carBrand;

   std::cout << "Array  size is " << sizeof(carBrand) << "\n";
   std::cout << "String size is " << strlen(carBrand) << "\n";
}
```

```
> ./CString3.exe

Please, enter your favorite car brand: BMW
Array  size is 10
String size is 3
```

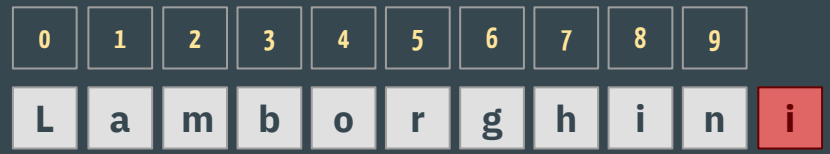
```
> ./CString3.exe

Please, enter your favorite car brand: Lamborghini
Segmentation fault
```

Выход за пределы массива

Работая с C-строками программист самостоятельно должен контролировать выход за пределы размера массива

```
char carBrand[10] = "";
```


```
std::cout << "Please, enter your favorite car brand: ";
std::cin >> carBrand; // Lamborghini
```




```
> ./CString3.exe

Please, enter your favorite car brand: Lamborghini
Segmentation fault
```

Преобразование строк

```
double atof(const char *string);
int atoi(const char *string);
long int atol(const char *string);
```

```
#include <stdlib.h>
#include <math.h>
#include <iostream>

const double PI = 3.1415926535;

int main ()
{
  char buffer[256];
  std::cout << "Enter degrees: ";
  std::cin >> buffer;

  double angle = atof(buffer);
  double result = sin(angle * PI / 180);
  std::cout <<"The sin of " << angle << " degrees is " << result << "\n";
}
```

```
> ./CString4.exe

Enter degrees: 45
The sin of 45 degrees is 0.707107
```

Копирование строк

```
char* strcpy (char* dst,
             const char* src);
```

```
#include <cstring>
#include <iostream>

int main ()
{
  char str1[128]="Plain C String";
  char str2[128];

  strcpy(str2, str1);
  strcpy(str1, "Another plain string");

  std::cout << "str1 = " << str1<< "\n";
  std::cout << "str2 = " << str2<< "\n";
  return 0;
}
```

```
> ./CString5.exe

str1 = Another plain string
str2 = Plain C String
```

Сравнение строк

```
int strcmp (const char* str1,
           const char* str2 );
```

```
const char PASSWORD[] = "a371_Z";

char userInput[128];

std::cout << "Please, enter password: ";
std::cin >> userInput;

const bool isSuccess = strcmp(PASSWORD, userInput) == 0;
```

Объединение строк

char* strcat (char* destination,
const char * source);

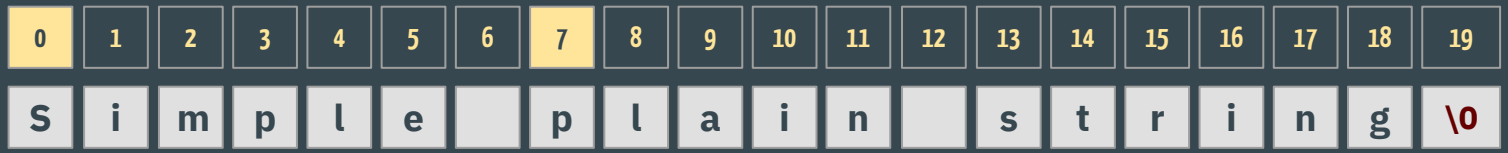
```
#include <cstring>
#include <iostream>

int main()
{
	char resultStr[255];
	char funcName[] = "strcat";

	strcpy(resultStr, funcName);
	strcat(resultStr, " function");
	strcat(resultStr, " concatenate strings");

	std::cout << "Result string size = " << strlen(resultStr) << "\n";
	std::cout << resultStr << "\n";
}
```

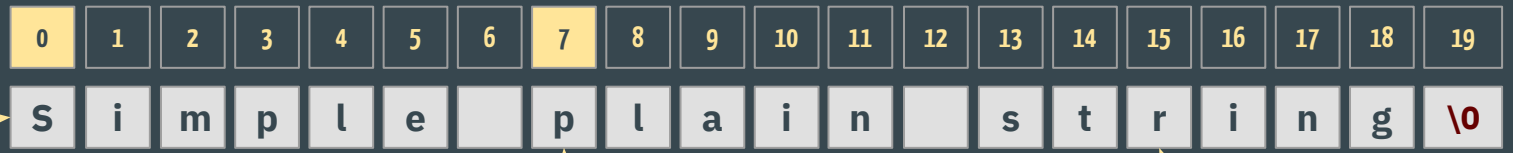
```
> ./CString7.exe

Result string size = 35
strcat function concatenate strings
```

Поиск подстроки

```
char* strstr(char* str1,
             const char* str2 );
```

```
char sourceStr[] = "Simple plain string";

char* foundStr = strstr(sourceStr, "plain");
if(foundStr  != NULL)
{
        const int foundPos = foundStr - sourceStr + 1;
        std::cout << "Word has been found in source string\n";
        std::cout << "It started from " << foundPos << " position\n";

        std::cout << "Print result: " << foundStr << "\n";
}
else
{
        std::cout << "Substring hasn't found in source string";
}
```






```
char sourceStr[] = "Simple plain string";

char* pStr1 = strstr(sourceStr, "plain");

char* pStr2 = strstr(sourceStr, "rain");

char* pStr3 = sourceStr + 15;

std::cout << "Print result: " << pStr3 << "\n";
```

NULL

Си строки. Практическое задание

1) Написать функцию копирования строки (Собственная реализация strcpy)
2) Написать функцию объединения строк (Собственная реализация strcat)
3) Написать функцию инвертирования строки