

# Увод в програмирането

Символни низове. Работа с текст

ФМИ, специалност „Софтуерно инженерство“

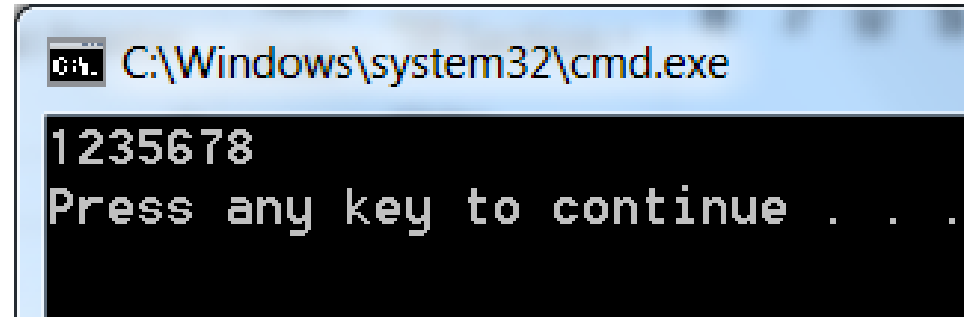
В тази презентация са използвани слайдове на  
доц. Атанас Семерджиев

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	<b>NUL</b> (null)	32	20	040	&#32;	<b>Space</b>	64	40	100	&#64;	<b>@</b>	96	60	140	&#96;	<b>`</b>
1	1	001	<b>SOH</b> (start of heading)	33	21	041	&#33;	<b>!</b>	65	41	101	&#65;	<b>A</b>	97	61	141	&#97;	<b>a</b>
2	2	002	<b>STX</b> (start of text)	34	22	042	&#34;	<b>"</b>	66	42	102	&#66;	<b>B</b>	98	62	142	&#98;	<b>b</b>
3	3	003	<b>ETX</b> (end of text)	35	23	043	&#35;	<b>#</b>	67	43	103	&#67;	<b>C</b>	99	63	143	&#99;	<b>c</b>
4	4	004	<b>EOT</b> (end of transmission)	36	24	044	&#36;	<b>\$</b>	68	44	104	&#68;	<b>D</b>	100	64	144	&#100;	<b>d</b>
5	5	005	<b>ENQ</b> (enquiry)	37	25	045	&#37;	<b>%</b>	69	45	105	&#69;	<b>E</b>	101	65	145	&#101;	<b>e</b>
6	6	006	<b>ACK</b> (acknowledge)	38	26	046	&#38;	<b>&amp;</b>	70	46	106	&#70;	<b>F</b>	102	66	146	&#102;	<b>f</b>
7	7	007	<b>BEL</b> (bell)	39	27	047	&#39;	<b>'</b>	71	47	107	&#71;	<b>G</b>	103	67	147	&#103;	<b>g</b>
8	8	010	<b>BS</b> (backspace)	40	28	050	&#40;	<b>(</b>	72	48	110	&#72;	<b>H</b>	104	68	150	&#104;	<b>h</b>
9	9	011	<b>TAB</b> (horizontal tab)	41	29	051	&#41;	<b>)</b>	73	49	111	&#73;	<b>I</b>	105	69	151	&#105;	<b>i</b>
10	A	012	<b>LF</b> (NL line feed, new line)	42	2A	052	&#42;	<b>*</b>	74	4A	112	&#74;	<b>J</b>	106	6A	152	&#106;	<b>j</b>
11	B	013	<b>VT</b> (vertical tab)	43	2B	053	&#43;	<b>+</b>	75	4B	113	&#75;	<b>K</b>	107	6B	153	&#107;	<b>k</b>
12	C	014	<b>FF</b> (NP form feed, new page)	44	2C	054	&#44;	<b>,</b>	76	4C	114	&#76;	<b>L</b>	108	6C	154	&#108;	<b>l</b>
13	D	015	<b>CR</b> (carriage return)	45	2D	055	&#45;	<b>-</b>	77	4D	115	&#77;	<b>M</b>	109	6D	155	&#109;	<b>m</b>
14	E	016	<b>SO</b> (shift out)	46	2E	056	&#46;	<b>.</b>	78	4E	116	&#78;	<b>N</b>	110	6E	156	&#110;	<b>n</b>
15	F	017	<b>SI</b> (shift in)	47	2F	057	&#47;	<b>/</b>	79	4F	117	&#79;	<b>O</b>	111	6F	157	&#111;	<b>o</b>
16	10	020	<b>DLE</b> (data link escape)	48	30	060	&#48;	<b>0</b>	80	50	120	&#80;	<b>P</b>	112	70	160	&#112;	<b>p</b>
17	11	021	<b>DC1</b> (device control 1)	49	31	061	&#49;	<b>1</b>	81	51	121	&#81;	<b>Q</b>	113	71	161	&#113;	<b>q</b>
18	12	022	<b>DC2</b> (device control 2)	50	32	062	&#50;	<b>2</b>	82	52	122	&#82;	<b>R</b>	114	72	162	&#114;	<b>r</b>
19	13	023	<b>DC3</b> (device control 3)	51	33	063	&#51;	<b>3</b>	83	53	123	&#83;	<b>S</b>	115	73	163	&#115;	<b>s</b>
20	14	024	<b>DC4</b> (device control 4)	52	34	064	&#52;	<b>4</b>	84	54	124	&#84;	<b>T</b>	116	74	164	&#116;	<b>t</b>
21	15	025	<b>NAK</b> (negative acknowledge)	53	35	065	&#53;	<b>5</b>	85	55	125	&#85;	<b>U</b>	117	75	165	&#117;	<b>u</b>
22	16	026	<b>SYN</b> (synchronous idle)	54	36	066	&#54;	<b>6</b>	86	56	126	&#86;	<b>V</b>	118	76	166	&#118;	<b>v</b>
23	17	027	<b>ETB</b> (end of trans. block)	55	37	067	&#55;	<b>7</b>	87	57	127	&#87;	<b>W</b>	119	77	167	&#119;	<b>w</b>
24	18	030	<b>CAN</b> (cancel)	56	38	070	&#56;	<b>8</b>	88	58	130	&#88;	<b>X</b>	120	78	170	&#120;	<b>x</b>
25	19	031	<b>EM</b> (end of medium)	57	39	071	&#57;	<b>9</b>	89	59	131	&#89;	<b>Y</b>	121	79	171	&#121;	<b>y</b>
26	1A	032	<b>SUB</b> (substitute)	58	3A	072	&#58;	<b>:</b>	90	5A	132	&#90;	<b>Z</b>	122	7A	172	&#122;	<b>z</b>
27	1B	033	<b>ESC</b> (escape)	59	3B	073	&#59;	<b>;</b>	91	5B	133	&#91;	<b>[</b>	123	7B	173	&#123;	<b>{</b>
28	1C	034	<b>FS</b> (file separator)	60	3C	074	&#60;	<b>&lt;</b>	92	5C	134	&#92;	<b>\</b>	124	7C	174	&#124;	<b> </b>
29	1D	035	<b>GS</b> (group separator)	61	3D	075	&#61;	<b>=</b>	93	5D	135	&#93;	<b>]</b>	125	7D	175	&#125;	<b>}</b>
30	1E	036	<b>RS</b> (record separator)	62	3E	076	&#62;	<b>&gt;</b>	94	5E	136	&#94;	<b>^</b>	126	7E	176	&#126;	<b>~</b>
31	1F	037	<b>US</b> (unit separator)	63	3F	077	&#63;	<b>?</b>	95	5F	137	&#95;	<b>_</b>	127	7F	177	&#127;	<b>DEL</b>

Source: [www.LookupTables.com](http://www.LookupTables.com)

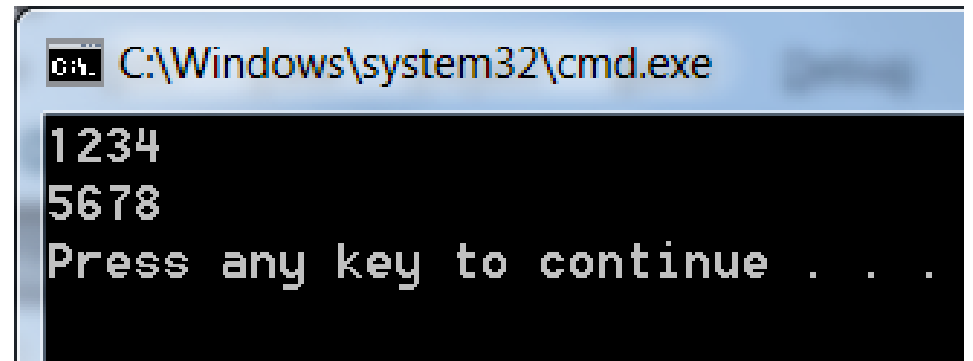
\* Източник на ASCII таблицата: <http://www.asciitable.com/>

```
int main()
{
    char a = 8;
    cout << "1234" << '\a' << "5678" << endl;
    return 0;
}
```



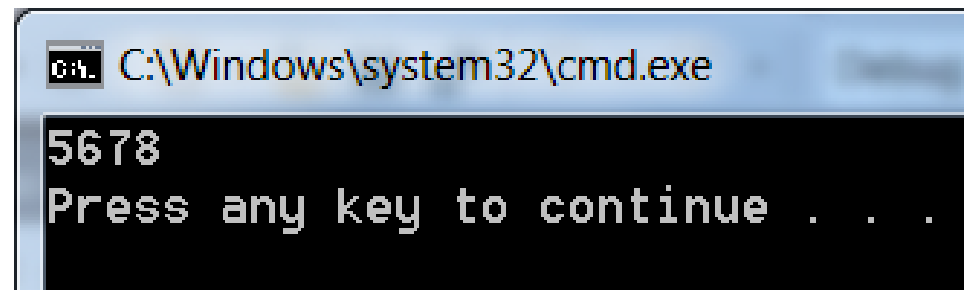
```
C:\Windows\system32\cmd.exe
1235678
Press any key to continue . . .
```

```
int main()
{
    cout << "1234" << '\n' << "5678" << endl;
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
1234
5678
Press any key to continue . . .
```

```
int main()
{
    cout << "1234" << '\r' << "5678" << endl;
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
5678
Press any key to continue . . .
```

# Escape Codes

Литерали за някои по използвани символи, които нямат графично представяне:

Тип	Описание
\n	Нов ред
\r	Carriage return
\b	Връщане на символ (backspace)
\t	Табулация
\a	Аларма/Сигнал

# Преобразуване до число/цифра

```
int ToInt(char c)
{
    if (c >= '0' && c <= '9')
        return c - '0';

    return 0;
}
```

```
int ToIntBitwise(char c)
{
    if (c >= '0' && c <= '9')
        return c & 0xF;

    return 0;
}
```

# Основни побитови операции

$$\begin{array}{r} \sim 1101 \\ \hline 0010 \end{array}$$

$$\begin{array}{r} | 1100 \\ 1010 \\ \hline 1110 \end{array}$$

$$\begin{array}{r} \& 1100 \\ 1010 \\ \hline 1000 \end{array}$$

$$\begin{array}{r} \wedge 1100 \\ 1010 \\ \hline 0110 \end{array}$$

# Преобразуване на регистър (letter case)

**Към горен регистър (uppercase)**

```
char ToUpper(char c)
{
    if (c >= 'a' && c <= 'z')
        return c - ('a' - 'A');

    return c;
}
```

**Към долен регистър (lowercase)**

```
char ToLower(char c)
{
    if (c >= 'A' && c <= 'Z')
        return c + ('a' - 'A');

    return c;
}
```

# Преобразуване чрез побитови операции

'A'	(65)	0	1	0	0	0	0	0	1
'a'	(97)	0	1	1	0	0	0	0	1

```
// 0x20 <--> 0010 0000  
char upper = 'A';  
char lower = upper | 0x20;
```

```
// 0xDF <--> 1101 1111  
char lower = 'a';  
char upper = lower & 0xDF;
```



# Преобразуване на регистър (letter case)

**Към горен регистър (uppercase)**

```
char ToUpperBitwise(char c)
{
    if (c >= 'a' && c <= 'z')
        return c & 0xDF;

    return c;
}
```

**Към долен регистър (lowercase)**

```
char ToLowerBitwise(char c)
{
    if (c >= 'A' && c <= 'Z')
        return c | 0x20;

    return c;
}
```

# Символни низове

- Последователност от символи
  - Последователност от 0 символа се нарича празен низ
- Представяне в C++: Масив от символи (char), в който след последния символ в низа е записан т.нар. терминиращ символ '\0'
  - '\0' е първият символ в ASCII таблицата, с код 0

# Включване на специални символи в низови литерали

\'	Апостроф
\"	Кавичка
\?	Питанка
\\	Обратна наклонена черта

// Литерали – символи

char Symbol1 = 'a'; // Малка латинска буква 'a'

char Symbol2 = '\n'; // Нов ред

char Symbol3 = '\x61'; // Код на символ в hex  
// Отново буквата 'a'

char Symbol4 = '\141'; // Код на символ в oct  
// Отново буквата 'a'

```
char Symbol5 = '''; // НЕВАЛИДНО!!!
```

```
char Symbol6 = '\\''; // Коректно
```

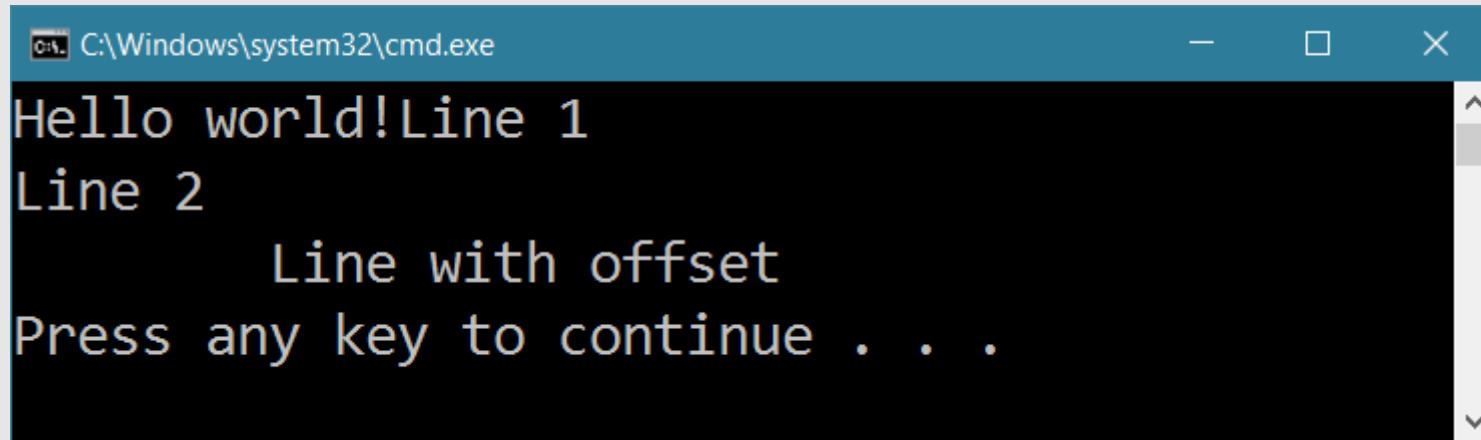
```
char Symbol7 = '\"'; // С кавичката  
// няма проблем
```

```
char Symbol8 = '\\\\'; // Обратна  
// наклонена черта
```

// Литерали – символни низове

```
cout << "Hello world!";
```

```
cout << "Line 1\nLine 2\n\tLine with offset\n";
```



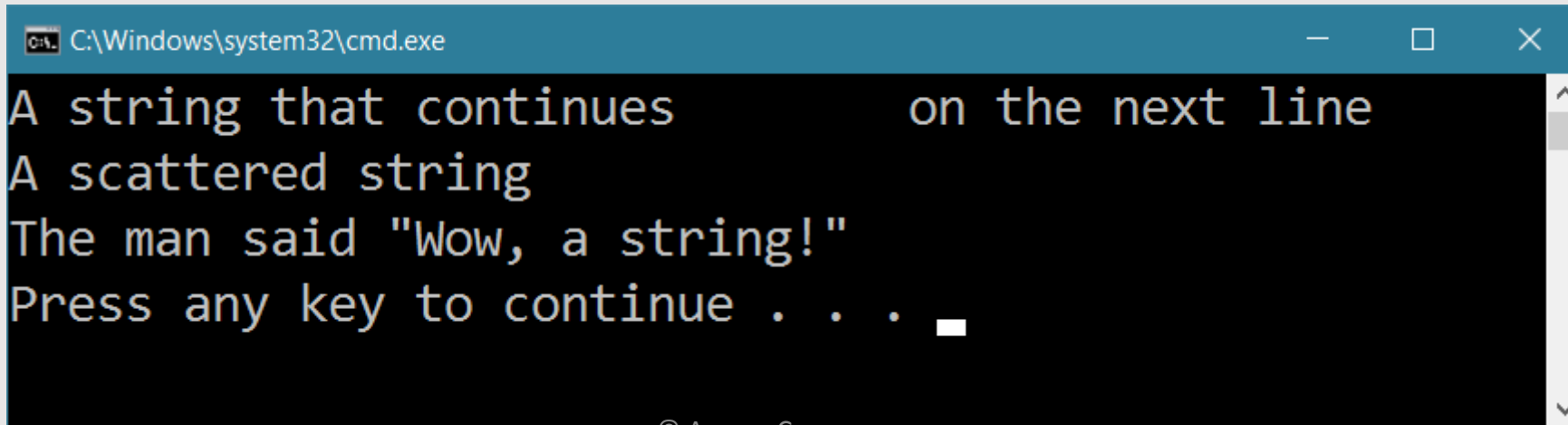
A screenshot of a Windows command prompt window. The title bar shows the path "C:\Windows\system32\cmd.exe". The window has a black background with white text. The output of the C++ program is displayed as follows: "Hello world!Line 1" on the first line, "Line 2" on the second line, and "Line with offset" on the third line, which is indented with a tab character. Below the output, the text "Press any key to continue . . ." is shown. A vertical scrollbar is visible on the right side of the window.

```
C:\Windows\system32\cmd.exe
Hello world!Line 1
Line 2
    Line with offset
Press any key to continue . . .
```

```
cout << "A string that continues \  
on the next line\n";
```

```
cout << "A " "scattered "  
"string\n";
```

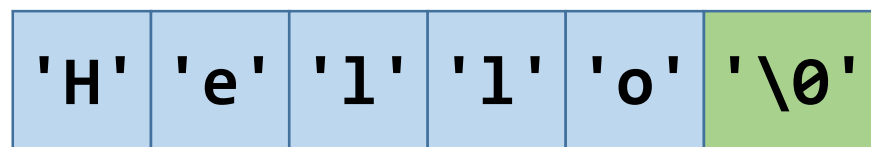
```
cout << "The man said \"Wow, a string!\"\n";
```



A screenshot of a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The window has a blue title bar with standard Windows window controls (minimize, maximize, close). The command prompt shows the output of the C++ code: "A string that continues" followed by "on the next line" on the next line, "A scattered string" on the next line, and "The man said \"Wow, a string!\"" on the next line. Below the output, it says "Press any key to continue . . ." with a white cursor. The background of the command prompt is black, and the text is white.

# Представяне на низ в паметта

```
char word[] = "Hello";
```



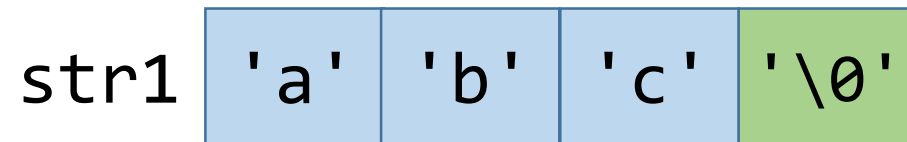
Текстът съдържа 5 символа  
Представя се чрез 6 символа

*Терминиращ  
елемент*

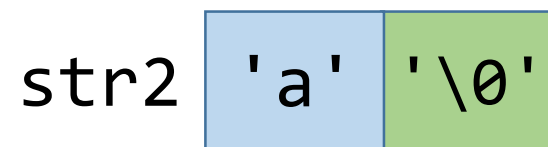


# Представяне на низ в паметта

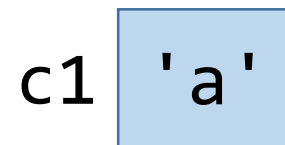
```
char str1[] = "abc";
```



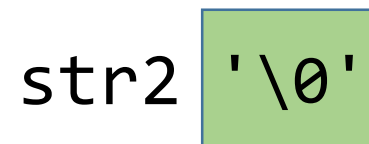
```
char str2[] = "a";
```



```
char c1 = 'a';
```



```
char str3[] = "";
```



```
char c2 = ' '; // Грешка!
```

```
int main()
{
    char word_1[] = { 'H', 'e', 'l', 'l', 'o', '\0' };
    char word_2[6] = { 'H', 'e', 'l', 'l', 'o' };
    char word_3[100] = "Hello";
    // char word_4[5] = "Hello"; Грешно!!!
    char word_5[6] = "Hello";
    char word_6[5] = { 'H', 'e', 'l', 'l', 'o' };

    cout << "word_1 is: " << word_1 << endl;
    cout << "word_2 is: " << word_2 << endl;
    cout << "word_3 is: " << word_3 << endl;
    cout << "word_5 is: " << word_5 << endl;
    cout << "word_6 is: " << word_6 << endl;

    return 0;
}
```

cmd C:\Windows\system32\cmd.exe

word\_1 is: Hello

word\_2 is: Hello

word\_3 is: Hello

word\_5 is: Hello

word\_6 is: Hello||||||||||||||||Hello

Press any key to continue . . . \_

# ВАЖНО!

Между следните има разлика:

'a' – число, кодът на буквата *a*.

"a" – символен низ (масив с два елемента – код на буква и терминиращ символ).

Респективно дадените по-долу редове се изпълняват различно:

```
std::cout << 'a';
```

```
std::cout << "a";
```

# Важно!

- Стринговете са масиви и затова НЕ МОЖЕ:
  - Да ги копираме с оператора за присвояване (=)
  - Да ги сравняваме с оператора за сравнение (==)
- За тези цели можем да използваме специално подготвени за целта библиотечни функции.

```
char str[] = "Abc";  
char buffer[100];
```

```
buffer = str;
```

```
if (buffer == str)  
{  
    // ...something...  
}
```

*Няма смисъл!!!*

# Намиране на дължина

```
size_t strlen(const char* str)
{
    if (str == nullptr) return -1;

    const char* pRead = str;

    while (*pRead != '\0')
        pRead++;

    return pRead - str;
}
```

# Намиране на дължина

```
size_t strlen(const char* str)
{
    if (str == nullptr) return -1;

    const char* pRead;

    for (pRead = str; *pRead != '\0'; pRead++)
        ;

    return str - pRead;
}
```

# Копиране на низ

```
char * strcpy(char* dest, const char* src)
{
    if (dest == nullptr) return nullptr;
    if (src == nullptr) return dest;

    while (*src != '\0')
    {
        *dest++ = *src++;
    }

    return dest;
}
```



# Пример за коректна реализация

```
char * strcpy(char* dest, const char* src)
{
    if (dest == nullptr) return nullptr;
    if (src == nullptr) return dest;

    while ((*dest++ = *src++) != '\0')
        ;

    return dest;
}
```

# Копиране на низ

```
char * strcpy(char* dest, const char* src)
{
    if (dest == nullptr) return nullptr;
    if (src == nullptr) return dest;

    do
    {
        *dest++ = *src;
    } while (*src++ != '\0');

    return dest;
}
```

# ВАЖНО!

Когато копирате един низ  $s_1$  в друг низ  $s_2$  се подсигурете, че:

1. В  $s_2$  има достатъчно място.
2. При копирането прехвърляте и терминиращата нула.

# Конкатенация

- Слепваме два низа един до друг
- За целта C++ предлага функцията `strcat`
- За резултантния низ трябва да са изпълнени:
  1. В него трябва да има достатъчно място.
  2. Той трябва да е правилно терминиран.

```
// Конкатенация на dest и src
char * strcat(char* dest, const char* src)
{
    if (dest == nullptr) return nullptr;
    if (src == nullptr) return dest;

    char *p = dest;

    while (*p != '\0')
        p++;

    strcpy(p, src);

    return dest;
}
```

# Сливане на няколко низа

```
char partA[] = "Hello";  
char partB[] = " ";  
char partC[] = "world!";
```

```
char buffer[100];
```

```
strcpy(buffer, partA);  
strcat(buffer, partB);  
strcat(buffer, partC);
```

# Сливане на няколко низа

```
char partA[] = "Hello";  
char partB[] = " ";  
char partC[] = " world!";
```

```
char buffer[100] = '\0';
```

```
strcat(buffer, partA);  
strcat(buffer, partB);  
strcat(buffer, partC);
```

// Конкатенация със слепващ елемент

```
char partA[] = "Hello";  
char partB[] = "from";  
char partC[] = "FMI!";
```

```
char buffer[100];
```

```
strcpy(buffer, partA);  
strcat(buffer, " ");  
strcat(buffer, partB);  
strcat(buffer, " ");  
strcat(buffer, partC);
```



# Важно!

- Винаги когато извършвате операции с низове се подsigурявайте, че:
  - Завършват с терминираща нула
  - При копиране в целевия масив има достатъчно място, включително и за терминиращата нула!

```
char str[] = "Hello world!";  
char buffer[5];  
strcpy(buffer, str);
```

*Грешка!!!*

# Неправилно използване на strcpy

```
void MyFunction(const char* Text)
{
    char Buffer[100];

    // Грешка: Не знаем, дали Text не съдържа
    // повече от 100 елемента!
    strcpy(Buffer, Text);

    //...
}
```

# Вариант 1: strcpy\_s

```
void MyFunction(const char* Text)
{
    char Buffer[100];

    // strcpy_s връща нула при успех
    if (strcpy_s(Buffer, 100, Text))
        // Обработваме грешката

    //...
}
```

## Вариант 2: strncpy

```
const int STR_SIZE;  
void MyFunction(const char* Text)  
{  
    char Buffer[STR_SIZE];  
  
    size_t Length = strlen(Text);  
  
    strncpy(Buffer, Text, min(Length, STR_SIZE-1))  
  
    //...  
}
```

## Вариант 3: външни проверки

```
void MyFunction(const char* Text)
{
    char Buffer[100];

    size_t Length = strlen(Text);

    if(Length >= 100)
        // Обработваме грешката
    else
        strcpy(Buffer, Text)

    //...
}
```

# Лексикографска наредба

AAA < BBB

ABC < ABD

ABC < ABCD

ABC ≠ abc

ABC < abc

# Лексикографска наредба

Дадени са два низа:

$$\begin{aligned} A &= a_1 a_2 \dots a_n \\ B &= b_1 b \dots b_m \end{aligned}$$

Имаме, че  $A < B$ , т.с.т.к е изпълнено едно от следните:

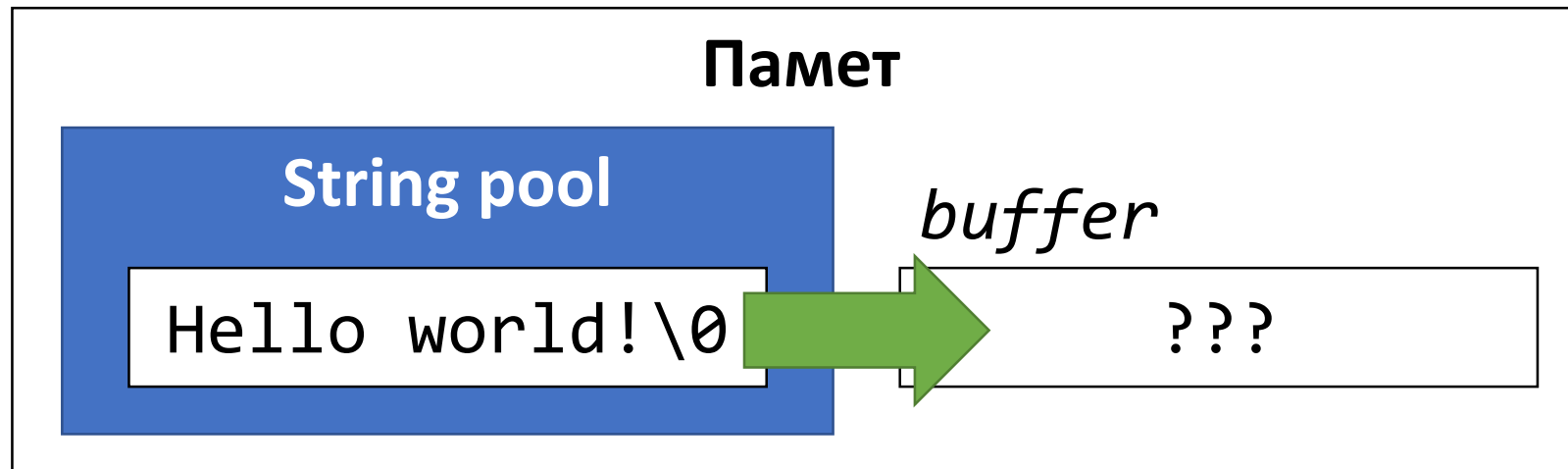
$$\begin{aligned} \exists i \left( i \leq \min(n, m) \wedge a_i < b_i \wedge \forall j < i (a_j = b_j) \right) \\ n < m \wedge \forall i \leq n (a_i = b_i) \end{aligned}$$

# Низови литерали



```
#include <string.h>

int main()
{
    char buffer[100];
    strcpy(buffer, "Hello world!");
}
```



# Валиден код

```
#include <iostream>

void main()
{
    char str1[] = "Abc";
    char str2[] = "Abc";

    str1[0] = 'a';

    std::cout << str1 << endl;
    std::cout << str2 << endl;
}
```

## Памет

*str1*

Abc\0

*str2*

Abc\0

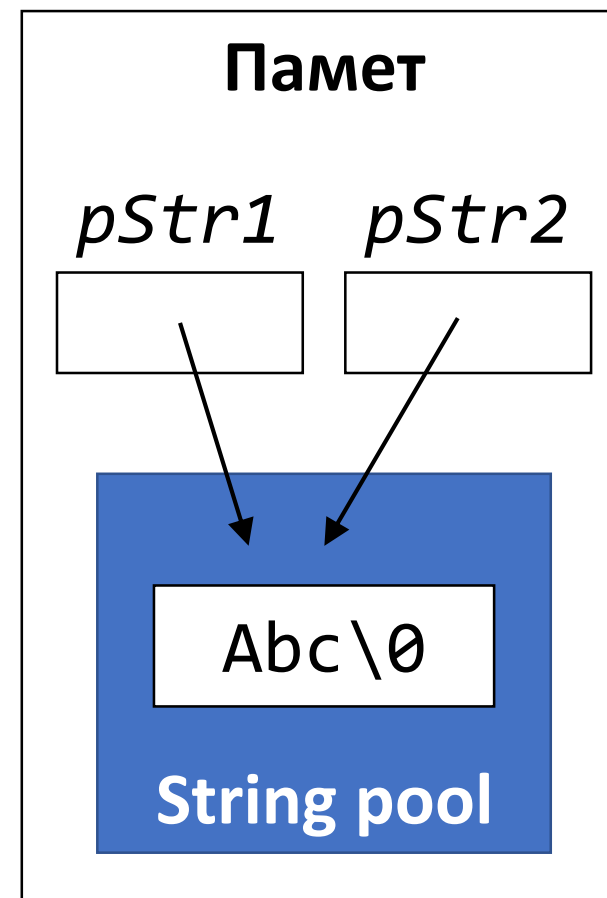
# Некоректен код

```
#include <iostream>

int main()
{
    char *pStr1 = "Abc";
    char *pStr2 = "Abc";

    pStr1[0] = 'a';

    std::cout << pStr1 << std::endl;
    std::cout << pStr2 << std::endl;
}
```



(\*Възможно представяне)