Увод в програмирането

12: Символни низове (стрингове)

доц. Атанас Семерджиев

Съдържание

- Работа с текст. Основни принципи.
- Символи
- Символни низове (стрингове)
- Основни операции
- String Pool

2

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

Source: www.LookupTables.com

3

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

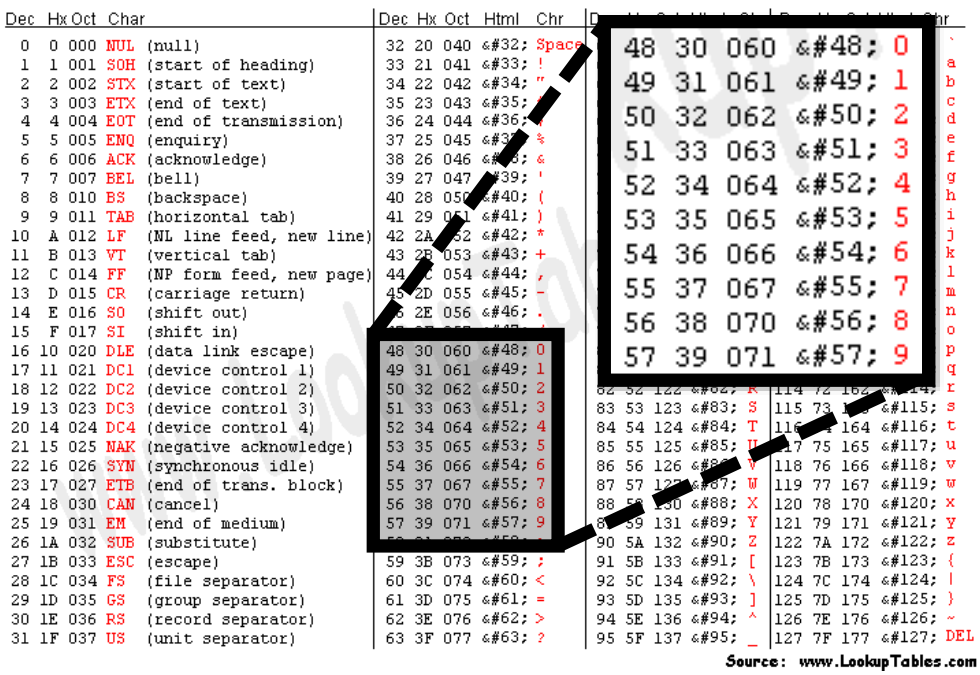
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of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

Source: www.LookupTables.com

7 7 007 BEL (bell)
8 8 010 BS (backspace)
9 9 011 TAB (horizontal tab)
10 A 012 LF (NL line feed, new line)
11 B 013 VT (vertical tab)
12 C 014 FF (NP form feed, new page)
13 D 015 CR (carriage return)

4

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



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space
1	1	001	SOH (start of heading)	33	21	041	!	!
2	2	002	STX (start of text)	34	22	042	"	"
3	3	003	ETX (end of text)	35	23	043	#	#
4	4	004	EOT (end of transmission)	36	24	044	$	$
5	5	005	ENQ (enquiry)	37	25	045	%	%
6	6	006	ACK (acknowledge)	38	26	046	&	&
7	7	007	BEL (bell)	39	27	047	'	'
8	8	010	BS (backspace)	40	28	050	(	(
9	9	011	TAB (horizontal tab)	41	29	051	)	)
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*
11	B	013	VT (vertical tab)	43	2B	053	+	+
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,
13	D	015	CR (carriage return)	45	2D	055	-	-
14	E	016	SO (shift out)	46	2E	056	.	.
15	F	017	SI (shift in)	47	2F	057	/	/
16	10	020	DLE (data link escape)	48	30	060	0	0
17	11	021	DC1 (device control 1)	49	31	061	1	1
18	12	022	DC2 (device control 2)	50	32	062	2	2
19	13	023	DC3 (device control 3)	51	33	063	3	3
20	14	024	DC4 (device control 4)	52	34	064	4	4
21	15	025	NAK (negative acknowledge)	53	35	065	5	5
22	16	026	SYN (synchronous idle)	54	36	066	6	6
23	17	027	ETB (end of trans. block)	55	37	067	7	7
24	18	030	CAN (cancel)	56	38	070	8	8
25	19	031	EM (end of medium)	57	39	071	9	9
26	1A	032	SUB (substitute)	58	3A	072	:	:
27	1B	033	ESC (escape)	59	3B	073	;	;
28	1C	034	FS (file separator)	60	3C	074	<	<
29	1D	035	GS (group separator)	61	3D	075	=	=
30	1E	036	RS (record separator)	62	3E	076	>	>
31	1F	037	US (unit separator)	63	3F	077	?	?

Source: www.LookupTables.com

5

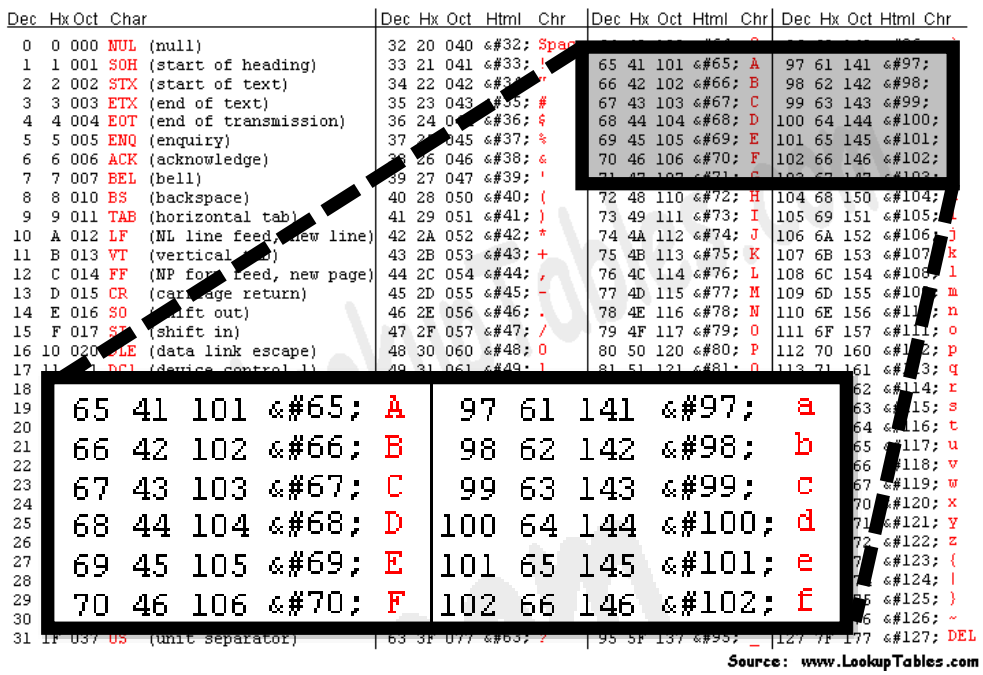
* Източник на ASCII таблицата: http://www.asciitable.com/

Преобразуване до число

```
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;
}
```

6



Dec	Hx	Oct	Char
0	0	000	NUL (null)
1	1	001	SOH (start of heading)
2	2	002	STX (start of text)
3	3	003	ETX (end of text)
4	4	004	EOT (end of transmission)
5	5	005	ENQ (enquiry)
6	6	006	ACK (acknowledge)
7	7	007	BEL (bell)
8	8	010	BS (backspace)
9	9	011	TAB (horizontal tab)
10	A	012	LF (NL line feed, new line)
11	B	013	VT (vertical tab)
12	C	014	FF (NP form feed, new page)
13	D	015	CR (carriage return)
14	E	016	SO (shift out)
15	F	017	SI (shift in)
16	10	020	DLE (data link escape)
17	11	021	DC1 (device control 1)
18	12	022	DC2 (device control 2)
19	13	023	DC3 (device control 3)
20	14	024	DC4 (device control 4)
21	15	025	NAK (negative acknowledge)
22	16	026	SYN (synchronous idle)
23	17	027	ETB (end of transmission block)
24	18	030	CAN (cancel)
25	19	031	EM (end of medium)
26	1A	032	SUB (substitute)
27	1B	033	ESC (escape)
28	1C	034	FS (file separator)
29	1D	035	GS (group separator)
30	1E	036	RS (record separator)
31	1F	037	US (unit separator)

Dec	Hx	Oct	Html	Chr
32	20	040		Space
33	21	041	!	!
34	22	042	"	"
35	23	043	#	#
36	24	044	$	$
37	25	045	%	%
38	26	046	&	&
39	27	047	'	'
40	28	050	(	(
41	29	051	)	)
42	2A	052	*	*
43	2B	053	+	+
44	2C	054	,	,
45	2D	055	-	-
46	2E	056	.	.
47	2F	057	/	/
48	30	060	0	0
49	31	061	1	1
63	3F	077	?	?
65	41	101	A	A
66	42	102	B	B
67	43	103	C	C
68	44	104	D	D
69	45	105	E	E
70	46	106	F	F
71	47	107	G	G
72	48	110	H	H
73	49	111	I	I
74	4A	112	J	J
75	4B	113	K	K
76	4C	114	L	L
77	4D	115	M	M
78	4E	116	N	N
79	4F	117	O	O
80	50	120	P	P
81	51	121	Q	Q
95	5F	137	_	_
97	61	141	a	a
98	62	142	b	b
99	63	143	c	c
100	64	144	d	d
101	65	145	e	e
102	66	146	f	f
103	67	147	g	g
104	68	150	h	h
105	69	151	i	i
106	6A	152	j	j
107	6B	153	k	k
108	6C	154	l	l
109	6D	155	m	m
110	6E	156	n	n
111	6F	157	o	o
112	70	160	p	p
113	71	161	q	q
114	72	162	r	r
115	73	163	s	s
116	74	164	t	t
117	75	165	u	u
118	76	166	v	v
119	77	167	w	w
120	78	170	x	x
121	79	171	y	y
122	7A	172	z	z
123	7B	173	{	{
124	7C	174			|
125	7D	175	}	}
126	7E	176	~	~
127	7F	177	&#127;	DEL

Source: www.LookupTables.com

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

7

Преобразуване на регистър (letter case)

Към горен регистър (uppercase)

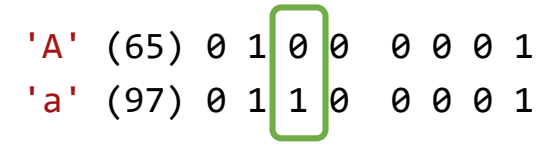
```
char ToUpper(char c)
{
   if (c >= 'a' && c <= 'z')
      return c - ('a' - 'A');

   return c;
}
```

Към долен регистър (lowercase)

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

   return c;
}
```

8

Преобразуване чрез побитови операции




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

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

9

Преобразуване на регистър (letter case)

Към горен регистър (uppercase)

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

   return c;
}
```

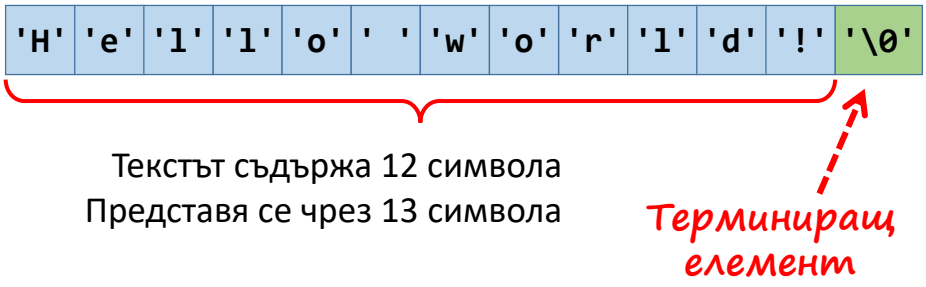
Към долен регистър (lowercase)

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

   return c;
}
```

10

Представяне на низ в паметта

```
char str[] = "Hello world!";
```



11

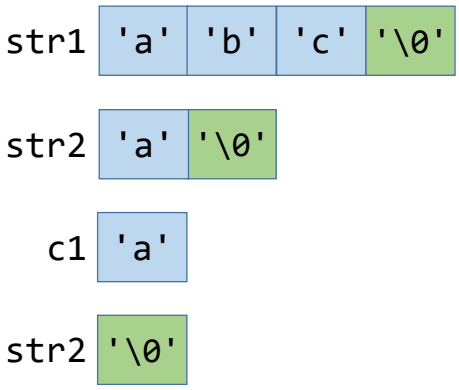
Създаване и инициализиране

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

// Еквивалентно
char str[] = { 'a','b','c','\0' };
```

12

Представяне на низ в паметта

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

char str2[] = "a";

char c1 = 'a';

char str3[] = "";

char c2 = ''; // Грешка!
```



13

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";
```

14

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...
}
```

Грешка!!!

15

Намиране на дължина

```
size_t strlen(const char* str)
{
    const char* pRead = str;

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

    return pRead - str;
}
```

16

Намиране на дължина

```
size_t strlen(const char* str)
{
    const char* pRead;

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

    return str - pRead;
}
```

17

Копиране на низ

```
char * strcpy(char* dest, const char* src)
{
    do
    {
        *dest++ = *src;
    } while (*src++ != '\0');

    return dest;
}
```

18

Пример за некоректна реализация

```
char * strcpy(char* dest, const char* src)
{
    while (*src != '\0')
    {
        *dest++ = *src++;
    }

    return dest;
}
```

19

Копиране на низ

```
char * strcpy(char* dest, const char* src)
{
    while ((*dest++ = *src++) != '\0')
        ;

    return dest;
}
```

20

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.

21

Конкатенация

- Слепваме два низа един до друг
- За целта C++ предлага функцията strcat
- За целевия низ трябва да са изпълнени:

1. В него трябва да има достатъчно място.
2. Той трябва да е правилно терминиран.

22

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

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

    strcpy(p, src);

    return dest;
}
```

23

Сливане на няколко низа

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

char buffer[100];

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

24

Сливане на няколко низа

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

char buffer[100];

buffer[0] = '\0';
strcat(buffer, partA);
strcat(buffer, partB);
strcat(buffer, partC);
```

25

```
// 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);
```

26

Лексикографска наредба

AAA < BBB

ABC < ABD

ABC < ABCD

ABC ≠ abc

ABC < abc

27

Лексикографска наредба

Дадени са два низа:

$$A = a_1 a_2 \dots a_n$$
$$B = b_1 b \dots b_m$$

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

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

28

```
int strcmp(const char *str1, const char *str2)
{
    unsigned char c1, c2;
    int diff;

    do {
        // str1 may be shorter and char is signed
        c1 = (unsigned char)*str1++;
        c2 = (unsigned char)*str2++;

        diff = c1 - c2;
    } while ((diff == 0) && (c1 != '\0'));

    return diff;
}
```

29

* http://research.microsoft.com/en-us/um/redmond/projects/invisible/src/crt/strcmp.c.htm

Важно!

- Винаги когато извършвате операции с низове се подсигурявайте, 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!!!

30

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);

    //...
}
```

31

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

    //...
}
```

32

Вариант 2: strncpy

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

    size_t Length = strlen(Text);

    strncpy(Buffer, Text, min(Length, 100-1))

    //...
}
```

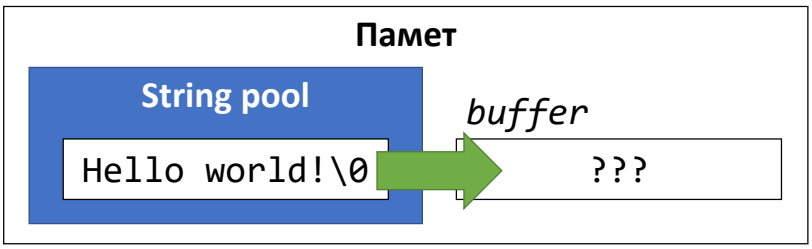
33

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)

    //...
}
```

34

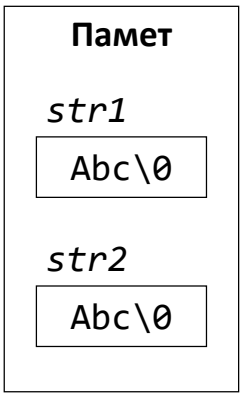
```
#include <string.h>

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



35

Валиден код

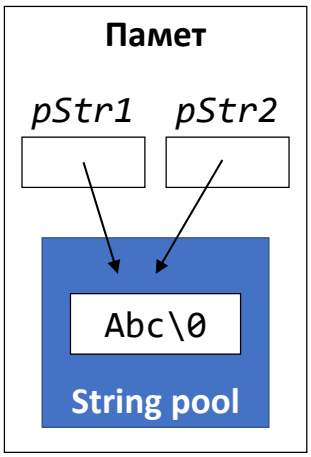
```
#include <iostream>

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

    str1[0] = 'a';

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



36

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)

37