



PROGRAMIRANJE I

P-09: Stringovi

prof. dr **Dražen Brđanin**
2023/24



P-09: Stringovi

- **Sadržaj predavanja**
 - definicija stringa
 - osnovne operacije nad stringovima
 - standardne funkcije (string.h)
 - o životnom vijeku stringova

Definicija stringa

String

- String je jednodimenzionalni niz znakova koji završava null-znakom (ASCII kod 0, tj. `'\0'`)
- Alternativno se stringovi nazivaju **nîske**, odnosno **niske završene nulom** (engl. *null-terminated strings*)

➤ Definicija stringa:

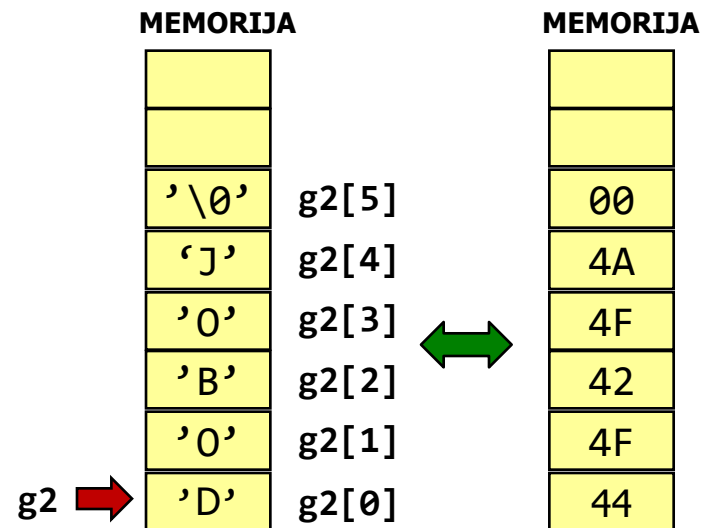
```
char ime[duzina]="string";  
char ime[]="string";  
char ime[duzina]={znak, znak, ... , 0};  
char ime[duzina]={znak, znak, ... , '\0'};
```

➤ Primjeri:

```
char g1[]="BANJA LUKA";  
char g2[]={ 'D', 'O', 'B', 'O', 'J', '\0' };
```

➤ Označavanje znakovnih i string konstanti:

```
'D' - ovo je znak D  
"D" - ovo je string { 'D', '\0' }
```



Neki programski jezici koriste drugačije konvencije za interni zapis stringova, npr. u Pascal-u se prije samog sadržaja stringa zapisuje njegova dužina (tzv. P-stringovi).

Osnovne operacije nad stringovima

Ispisivanje stringa

```
int printf(const char *format, ...)
```

- Formatirano ispisivanje stringa moguće je pozivom standardne funkcije **printf** (stdio.h), pri čemu se koristi konverzioni karakter %s

Primjer:

```
#include <stdio.h>
int main()
{
    char g[]="BANJA LUKA";
    printf("%s\n", g);
    printf("%15s\n", g);
    printf("%-15s\n", g);
    return 0;
}
```

```
BANJA LUKA
      BANJA LUKA
BANJA LUKA
```

```
int puts(const char *str)
```

- Funkcija **puts** (stdio.h) ispisuje string (bez terminatora), nakon čega prelazi u sljedeći red.

Primjer:

```
#include <stdio.h>
int main()
{
    char g1[]="BANJA";
    char g2[]="LUKA";
    puts(g1);
    puts(g2);
    return 0;
}
```

```
BANJA
LUKA
```

Osnovne operacije nad stringovima

Učitavanje stringa

```
int scanf(const char *format, ...)
```

- Formatirano učitavanje stringa moguće je pozivom standardne funkcije **scanf** (stdio.h), pri čemu se koristi konverzioni karakter %s

Primjer:

```
#include <stdio.h>
int main()
{
    char tekst[50];
    printf("Unesi tekst:\n");
    scanf("%s", tekst);
    printf("%s", tekst);
    return 0;
}
```

```
Unesi tekst:
BANJA LUKA
BANJA
```

```
char *gets(char *str)
```

- Funkcija **gets** (stdio.h) učitava string znak po znak sve do kraja reda (znak za kraj reda se ne upisuje u memoriju).

Primjer:

```
#include <stdio.h>
int main()
{
    char tekst[50];
    printf("Unesi tekst:\n");
    gets(tekst);
    printf("%s", tekst);
    printf("*");
    return 0;
}
```

```
Unesi tekst:
BANJA LUKA
BANJA LUKA*
```

Osnovne operacije nad stringovima

Određivanje dužine stringa

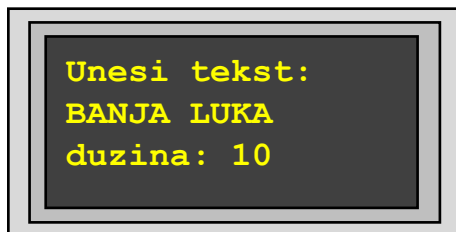
- String je niz znakova terminiran null-znakom, što pojednostavljuje implementaciju funkcija

Primjer:

```
#include <stdio.h>

int duzina(char *s)
{
    int i=0;
    while (*s++) i++;
    return i;
}
```

```
int main()
{
    char s[50];
    printf("Unesi tekst:\n");
    gets(s);
    printf("duzina: %d", duzina(s));
    return 0;
}
```



➤ Alternativna implementacija #1

```
int duzina(char s[])
{
    int i=0;
    while (s[i]) i++;
    return i;
}
```

➤ Alternativna implementacija #2

```
int duzina(const char *s)
{
    int i=0;
    while (*(s+i)!='\0') i++;
    return i;
}
```

➤ Korištenje standardne funkcije (string.h)

```
size_t strlen(const char *str)
```

Osnovne operacije nad stringovima

Lingvističko poređenje dva stringa

Primjer:

```
#include <stdio.h>

int cmp(const char *s1, const char *s2)
{
    while (*s1 && *s2 && *s1==*s2) s1++, s2++;
    return *s1-*s2;
}

int main()
{
    char s1[50], s2[50];
    printf("s1: "); gets(s1);
    printf("s2: "); gets(s2);

    int rez = cmp(s1,s2);
    if (rez<0) printf("%s < %s\n", s1, s2);
    else if (rez==0) printf("%s = %s\n", s1, s2);
    else printf("%s > %s\n", s1, s2);

    return 0;
}
```

```
s1: BANJA LUKA
s2: BEOGRAD
BANJA LUKA < BEOGRAD
```

```
s1: BEOGRAD
s2: BANJA LUKA
BEOGRAD > BANJA LUKA
```

```
s1: BANJA LUKA
s2: BANJA LUKA
BANJA LUKA = BANJA LUKA
```

➤ Korištenje standardne funkcije (string.h)

```
int strcmp(const char *s1, const char *s2)
```



Osnovne operacije nad stringovima

Kopiranje stringa

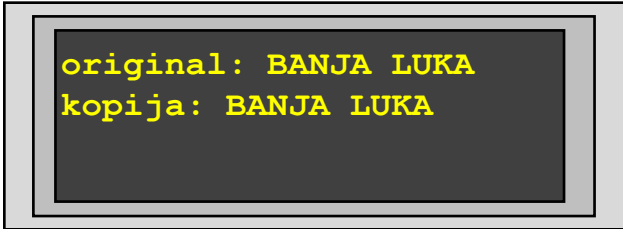
Primjer (kreiranje dinamičke kopije stringa):

```
#include <stdio.h>
#include <stdlib.h>

int duzina(const char *s)
{
    int i=0;
    while (*s++) i++;
    return i;
}

char *kopija (const char *s)
{
    int i = 0;
    // alokacija dinamičkog stringa
    char *rez = (char*) malloc(duzina(s)+1);
    if (!rez) return 0;
    while (*s) rez[i++]=*s++;    // kopiranje
    rez[i]=0; // terminiranje rezultata
    return rez;
}
```

```
int main()
{
    char t[50];
    printf("original: "); gets(t);
    char *s = kopija(t);
    if (s)
    {
        printf("kopija: %s\n", s);
        free(s);
    }
    else
        printf("Greaka!");
    return 0;
}
```



```
original: BANJA LUKA
kopija: BANJA LUKA
```




Osnovne operacije nad stringovima

Spajanje (konkatenacija) dva stringa

```
#include <stdio.h>
#include <stdlib.h>

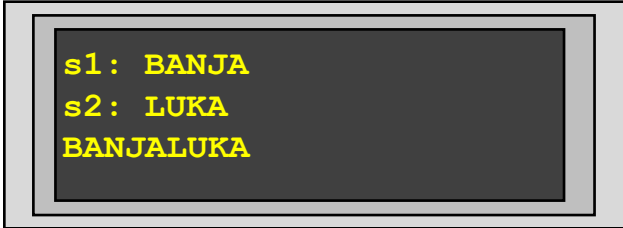
int duzina(const char *s)
{
    int i=0;
    while (*s++) i++;
    return i;
}

char *spajanje(const char *s1, const char *s2)
{
    int i = 0;
    // duzina rezultatnog stringa
    int d = duzina(s1) + duzina(s2) + 1;
    // alokacija dinamičkog stringa
    char *rez = (char*) calloc(d, sizeof(char));
    if (!rez) return 0;

    while (*s1) rez[i++] = *s1++; // kopiranje prvog
    while (*s2) rez[i++] = *s2++; // kopiranje drugog

    return rez;
}
```

```
int main()
{
    char s1[50], s2[50];
    printf("s1: "); gets(s1);
    printf("s2: "); gets(s2);
    char *s = spajanje(s1, s2);
    if (s)
    {
        printf("%s\n", s);
        free(s);
    }
    else
        printf("Greaka!");
    return 0;
}
```



```
s1: BANJA
s2: LUKA
BANJALUKA
```



Standardne funkcije (string.h)

<code>void *memcpy(void *dest, const void *src, size_t n)</code>	Kopira prvih n znakova iz <i>src</i> u <i>dest</i> .
<code>void *memset(void *str, int c, size_t n)</code>	Kopira znak c u prvih n znakova stringa na koji pokazuje argument <i>str</i> .
<code>char *strcat(char *dest, const char *src)</code>	Dodaje string na koji pokazuje <i>src</i> na kraj stringa na koji pokazuje <i>dest</i> .
<code>char *strchr(const char *str, int c)</code>	Traži prvo pojavljivanje znaka c u stringu na koji pokazuje <i>str</i> .
<code>char *strrchr(const char *str, int c)</code>	Traži posljednje pojavljivanje znaka c u stringu na koji pokazuje <i>str</i> .
<code>int strcmp(const char *s1, const char *s2)</code>	Poredi string na koji pokazuje <i>s1</i> sa stringom na koji pokazuje <i>s2</i> ($s1 < s2 \Rightarrow -1$ / $s1 == s2 \Rightarrow 0$ / $s1 > s2 \Rightarrow 1$).
<code>char *strcpy(char *dest, const char *src)</code>	Kopira string sa <i>src</i> na <i>dest</i> .
<code>size_t strlen(const char *str)</code>	Vraća dužinu stringa <i>str</i> bez nul znaka.
<code>char *strstr(const char *s, const char *p)</code>	Pronalazi prvo pojavljivanje podstringa p (bez terminatora) u stringu s .
...	

O životnom vijeku stringova

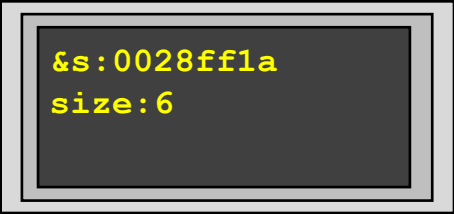
Automatski (STACK)

- lokalni automatski niz

Primjer:

```
#include <stdio.h>
#define S(x) sizeof(x)

int main()
{
    char s[]="STACK";
    printf("&s:%p\n", &s);
    printf("size:%d\n", S(s));
    return 0;
}
```



```
&s:0028ff1a
size:6
```

Primjer:

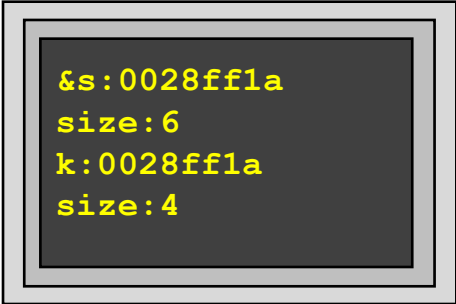
```
#include <stdio.h>
#define S(x) sizeof(x)

void f(char k[])
{
    printf("k:%p\n", k);
    printf("size:%d\n", S(k));
}

int main()
{
    char s[]="STACK";
    printf("&s:%p\n", &s);
    printf("size:%d\n", S(s));
    f(s);
    return 0;
}
```

s je smješten na steku
i zauzima 6 bajtova

k sadrži adresu od s i
zato je njegova
veličina 4



```
&s:0028ff1a
size:6
k:0028ff1a
size:4
```

O životnom vijeku stringova

Statički (DATA)

• globalni niz

Primjer:

```
#include <stdio.h>
#define S(x) sizeof(x)
char g[]="DATA";
int main()
{
    printf("&g:%p\n", &g);
    printf("size:%d\n", S(g));
    return 0;
}
```

```
&g:00407000
size:5
```

• lokalni statički niz

Primjer:

```
#include <stdio.h>
#define S(x) sizeof(x)
int main()
{
    static char s[]="DATA";
    printf("&s:%p\n", &s);
    printf("size:%d\n", S(s));
    return 0;
}
```

```
&s:00407000
size:5
```

• konstantni stringovi

Primjer:

```
#include <stdio.h>
#define S(x) sizeof(x)
int main()
{
    char *s="DATA";
    printf("&s:%p\n", &s);
    printf("s: %p\n", s);
    printf("size:%d\n", S(s));
    return 0;
}
```

```
&s:0028ff1c
s: 00408024
size:4
```

s je lokalni pokazivač (na steku) koji pokazuje na konstantni string u DATA seg.

O životnom vijeku stringova

Dinamički (HEAP)

- dinamički alociran niz

Primjer:

```
#include <stdio.h>
#include <stdlib.h>
#define S(x) sizeof(x)

int main()
{
    char *s = (char*) malloc(10);
    printf("&s:%p\n", &s);
    printf("s: %p\n", s);
    printf("size:%d\n", S(s));
    return 0;
}
```

s je lokalni pokazivač
(smješten na steku)
koji pokazuje na dinamički
alocirani string
(smješten u hip segmentu).

```
&s:0028ff1c
s: 008c0fe0
size:4
```