

PODATKI TIPA STRING

NIZ ali STRING prikažemo kot tabelo, ki je sestavljen iz več znakov, ter zaključen z ASCII kodo 0.

T	O	'	J	E	'	N	I	Z	'\0'																			
---	---	---	---	---	---	---	---	---	------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Splošne lastnosti:

- predstavitev podatkovnega tipa: kot vektor, kot tabela
- število segmentov določimo ob deklaraciji - indeks je tipa integer, celo število od 0 do n -1
- kot večina podatkovnih tipov, ima tudi string svoje standardne funkcije in procedure.
- Deklaracijske zakonitosti:
 - char niz[255];
 - dolžina niza je nepredznačeno celo število od 0 – 254

P	O	N	E	D	E	L	J	E	K
---	---	---	---	---	---	---	---	---	---

- niz je vektorska veličina, ima začetek, smer in dolžino.
- Vsak niz ima za podatki en blok, v katerem je zapisan konec niza – ascii koda 0!

P	O	N	E	D	E	L	J	E	K	\0
---	---	---	---	---	---	---	---	---	---	----

Zgled deklaracije:

```
int main(void)
{
    char a[20]; /* niz je sestavljen iz 20 znakov */
    return 0;
}
```

a - dvajset znakov

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

Deklaracija konstante nad podatkovnim tipom string;

Inicializacija ob deklaraciji in izpis podatkov:

```
int main(void)
{
    char a[20] = "Primer"; /* niz je sestavljen iz 20 znakov */
    printf("V spremenljivki a je vrednost %s\n",a);
    return 0;
}
```

nepravilna uporaba (napako javi že prevajalnik!)

```
int main(void)
{
    char a[20];
    a="Primer"; // vrednosti prirejamo s pomočjo vgrajenih funkcij
    printf("%s\n",a);
    return 0;
}
```

Branje podatkov:

```
int main(void)
{
    char a[20]; /* niz je sestavljen iz 20 znakov */
    printf("Vnesi niz: ");
    scanf("%s",a); /* ni znaka & pred spremenljivko*/
    printf("Vnesen niz je %s.\n",a);
    return 0;
}
```

Namesto scanf, gets uporabi raje fgets. Lahko pa se uporabi tudi scanf ali gets, samo paziti je potrebno na velikost vnešenega teksta – scanf in gets nimata kontrole vnosa števila podatkov in lahko pride do prepisa podatkov (overflow!)

Pri scanf se prebere samo ena beseda (torej do presledka!), gets in fgets pa prebere tudi znak za skok v novo vrstico. Torej vsak ima svoj problem. Lahko se uporabi tudi zanko in bere znak po znak z ukazom `getchar ()` vse dokler ne vnesemo znaka `'\n'` – le tega nato zamenjamo z `'\0'`.

```
#include <stdio.h>
int main(void)
{
    char p[200];
    printf("Vnesi niz: ");
    scanf("%s",p); // pazi prebere samo eno besedo!
    printf("Vnesen niz je %s.\n",p);
    fflush(stdin);
    printf("Vnesi niz: ");
    fgets(p,200,stdin); // prebere vse dokler ne pritisnemo ENTER – žal se tudi znak enter shrani
    printf("Vnesen niz je %s.\n",p);
    return 0;
}
```

Če v ta program vnesemo pri prvem vnosu "Kekec in pehta" in pri drugem vnosu "Kekec in pehta" nam bo program izpisal:

```
Vnesi niz: Kekec in pehta
Vnesen niz je Kekec.
Vnesi niz: Kekec in pehta
Vnesen niz je Kekec in pehta
.
```

Če je vnesena beseda daljša od velikosti niza, se ta odreže in na zadnjem znaku se vstavi znak za konec niza `'\0'`.

Kako se znebimo skoka v novo vrstico, če preberemo niz s pomočjo ukaza fgets

```
if (niz[strlen(b2)-1] == '\n')
{
    // vnesemo celotno vrstico
    niz[strlen(b2)-1] = '\0'; // zamenjamo skok v novo vrstico z 0 znakom
}
```

stdin, stdout, stderr, fflush

stdin – standard input – navadno tipkovnica

stdout – standard output – monitor

stderr – standard error – naprava na katero naj se prikazujejo napake – monitor

fflush – počisti - izprazni (izpiše, prebere) vse kar je ostalo od prejšnjih ukazov v tokovih (stdin,stdout,stderr).

NIZOVNE FUNKCIJE IN PROCEDURE

Za uporabo nizovnih funkcij potrebujemo vključiti knjižnico **string.h**

char *strcpy(const char *dest, const char *src) – Kopiraj niz iz enega v drugega, kazalec je postavljen na konec niza.

int strcmp(const char *string1, const char *string2) – Primerjaj dva niza - alfabetično

char *strcpy(const char *string1, const char *string2) – Kopira niz string2 v string1.

char *strerror(int errnum) – pridobi sporočilo o napaki, glede na napako

char *strcat(char *dest, const char *src) – pripni niz znakov iz src niza v dest niz

int strlen(const char *string) – Vrni dolžino nekega niza

char *strncat(const char *string1, const char *string2, size_t n) – pripni n znakov iz string2 na string1.

int strncmp(const char *string1, const char *string2, size_t n) – Primerjaj prvih n znakov dveh nizov.

char *strncpy(const char *string1, const char *string2, size_t n) – kopiraj prvih n znakov string2 v string1.

int strcasecmp(const char *s1, const char *s2) – ne glede na velikost črk primerjaj niza - kot strcmp().

int strncasecmp(const char *s1, const char *s2, int n) – ne glede na velikost črk primerjaj n znakov v nizih - kot strncmp().

Iskalne funkcije

char *strchr(const char *string, int c) – Najde prvi znak c v nizu string

char *strrchr(const char *string, int c) – Najde zadnji znak c v nizu string.

char *strstr(const char *s1, const char *s2) – najde prvi niz s2 v nizu s1.

char *strpbrk(const char *s1, const char *s2) -- returns a pointer to the first occurrence in string s1 of any character from string s2, or a null pointer if no character from s2 exists in s1

size_t strspn(const char *s1, const char *s2) -- returns the number of characters at the beginning of s1 that match s2.

size_t strcspn(const char *s1, const char *s2) -- returns the number of characters at the beginning of s1 that *do not* match s2.

char *strtok(char *s1, const char *s2) -- break the string pointed to by s1 into a sequence of tokens, each of which is delimited by one or more characters from the string pointed to by s2.

char *strtok_r(char *s1, const char *s2, char **lasts) -- has the same functionality as strtok() except that a pointer to a string placeholder lasts must be supplied by the caller.

Primeri uporabe funkcije:

strcpy

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

```
#include <string.h>
```

```
int main(void)
```

```
{
```

```
    char a[20];
```

```
    strcpy(a, "Primer");
```

```
    printf("%s\n", a); // Izpiše: Primer
```

```
    return 0;
```

```
}
```

strcmp

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

```
int main(void)
{
    char a[20];
    strcpy(a, "Primer");
    printf("%d\n", strcmp(a, "ime")); // Izpise: -1 - p za i
    printf("%d\n", strcmp("ime", a)); // Izpise: 1 - i pred p
    printf("%d\n", strcmp(a, a));    // Izpise: 0 - niza sta enaka
    printf("%s\n", a);
    return 0;
}
```

strcat

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

```
int main(void)
{
    char a[20];
    char b[20] = " za nize.";
    strcpy(a, "Primer");
    printf("%s\n", a);           // Izpise: Primer
    printf("%s\n", b);           // Izpise: za nize.
    printf("%s\n", strcat(a, b)); // Izpise: Primer za nize.
    printf("%s\n", a);           // Izpise: Primer za nize.
    return 0;
}
```

Pretvorba niza v število (atoi in atof)

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

```
int main(void)
{
    int n;
    float m;
    char str[20] = "12345.67";

    n = atoi(str);
    m = atof(str);
    printf("string = %s int = %d in float = %0.2f\n", str, n, m);
    return 0;
}
```

Pretvorba iz števila v niz (itoa, fcvt)

```
#include <stdlib.h>
#include <stdio.h>
int main(void)
{
    int number = 12345;
    char string[25];

    itoa(number, string, 10);
    printf("int = %d string = %s\n", number, string);
    return 0;
}

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main(void)
{
    char str[25];
    double num;
    int dec, sign, ndig = 5;

    /* realno stevilo */
    num = 9.876;
    strcpy(str, fcvt(num, ndig, &dec, &sign));
    printf("string = %s decimalna mesta = %d predznak = %d\n", str, dec, sign);
    /* izpis: string = 987600 decimalna mesta = 1 predznak = 0 */

    /* negativno realno stevilo */
    num = -123.45;
    strcpy(str, fcvt(num, ndig, &dec, &sign));
    printf("string = %s decimalna mesta = %d predznak = %d\n", str, dec, sign);
    /* izpis: string = 12345000 decimalna mesta = 3 predznak = 1 */

    /* realno stevilo zapisano na znanstveni način */
    num = 0.678e5;
    strcpy(str, fcvt(num, ndig, &dec, &sign));
    printf("string = %s decimalna mesta = %d predznak = %d\n", str, dec, sign);
    /* izpis: string = 6780000000 decimalna mesta = 5 predznak = 0 */

    return 0;
}
```

Uporaba **sprintf** za raznorazne pretvorbe. Namesto, da se rezultat izpiše na zaslon, se rezultat izpiše v spremenljivko.

```
#include <stdlib.h>
#include <stdio.h>
int main(void)
{
    float stevilo = 6.5126;
    char str[256];
    sprintf(str, "%.4f", stevilo);
    printf("%s\n", str); // izpiše nam 6.5126
    return 0;
}
```

Podprogrami in nizi

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

```
void beri(char *niz)
{
    printf("Vnesi niz: ");
    fgets(niz,256,stdin);
    // zamenjamo skok v novo vrstico z 0 znakom
    if (niz[strlen(niz)-1] == '\n')
    {
        niz[strlen(niz)-1] = '\0';
    }
}
```

```
void izpis(const char *niz)
{
    printf("%s",niz);
}
```

```
int main(void)
{
    char s[256];
    beri(s);
    izpis(s);
    return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int i;
int t[26];
```

```
void nastavi(int *t) {
    for (i=0;i<26;i++) t[i]=0;
}
```

```
void izpis(int *t) {
    for (i=0;i<26;i++)
        if (t[i] > 0) printf("%c - %d\n",i+65, t[i]);
}
```

```
void kaj(int *t, const char *x) {
    for (i=0; i < strlen(x); i++)
        t[x[i]-65]=t[x[i]-65]+1;
}
```

```
int main(void) {
    nastavi(t); kaj(t,"PLAVALKA"); izpis(t);
    nastavi(t); kaj(t,"TRST"); izpis(t);
    return 0;
}
```

Nizi v tabeli

Prikaz kako znotraj glavnega programa shranimo niz v tabelo.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(void)
{
    char p[5][200];
    int i;
    for (i=0;i<5;i++)
    {
        printf("Vnesi niz: ");
        fgets(p[i],200,stdin);
    }
    printf("vnesel si naslednje nize:\n");
    for (i=0;i<5;i++)
        printf("%d.niz: %s\n",i,p[i]);
    return 0;
}
```

Prikaz kako v podprogram vnesemo nize iz tabele

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
void beri(char *niz)
{
    printf("Vnesi niz: ");
    fgets(niz,256,stdin);
    if (niz[strlen(niz)-1] == '\n') { niz[strlen(niz)-1] = '\0'; }
}
void izpis(const char *niz) {
    printf("%s",niz);
}
int main(void) {
    char s[5][256];
    int i;
    for (i=0;i<3;i++) {
        beri(s[i]);
        izpis(s[i]); }
    return 0;
}
```

Naloge

- Izdelaj podprogram, ki izpiše dolžino niza (brez uporabe funkcije `strlen`). Primer glave `int length(const char* niz);`
- Izdelaj podprogram, ki bo vrnil podniz od določene pozicije za določeno število znakov. Primer glave `void strsub(const char *src,int pos, int length, char *dest);`
- Izdelaj podprogram, ki nam pretvori celo število v niz – potrebno je tudi paziti na dolžino števila! Primer glave `void inttostr(int a,char *dest,int n);`
- Izdelaj podprogram, ki nam pretvori niz v celo število, pri čemer je potrebno preverjati ali je število ustrezno (nima prepovedanih črk). Če slučajno naletimo na črko, podprogram vrne pozicijo črke, ter pretvorjeno število. Primer glave `int strtoint(const char *src,int *err);`
- Izdelaj podprogram, ki vrne prvih n znakov (zadnje znake odreže) iz določenega niza. Primer glave `void trim(const char *src, char *dest, int n);`
- Izdelaj podprogram, ki vse določene znake zamenja z drugim (primer vse 'e' zamenjaj z 'E'). Primer glave `void replace_char(char *src, char in, char out);`
- Izdelaj podprogram, ki vse določene znake izbriše iz niza (primer vse 'e' izbriše iz niza). Primer glave `void delete_char(char *src, char in);`
- Izdelaj podprogram, ki iz določene pozicije izbriše določeno število znakov (Iz besede "Primer" želimo izbrisati od vključno 3 znaka 'i', nadalnje 3 znake "ime"). Primer glave `void delete(char *src, int poz, int len);`
- Izdelaj podprogram, ki na določeno mesto vrine črko. Primer glave `void insert_char(char *dest, int i, char c);`
- Izdelaj podprogram, ki na določeno mesto vrine nov niz. Primer glave `void insert(char *dest, const char *src; int i);`