Uvod u programiranje

- predavanja -

prosinac 2025.

16. Pokazivači

Pokazivači

Polja, funkcije i pokazivači

Kako *funkciji* omogućiti pristup članovima polja?

- Polje se ne može koristiti kao argument/parametar funkcije
  - kako onda funkciji omogućiti pristup članovima polja koje je definirano u funkciji na pozivajućoj razini?
- Iskoristiti mogućnost pristupa članovima polja pomoću pokazivača
  - kao argument/parametar navesti **pokazivač** na prvi član polja
    - parametar će biti kopija argumenta, ali to ne smeta: kopija pokazivača će također pokazivati na prvi član tog polja
  - dodati i argument/parametar koji opisuje koliko članova polja ima
  - u funkciji koristiti pokazivač da bi se pristupilo članovima polja koje je definirano u funkciji na pozivajućoj razini
    - ako je `p` pokazivač na prvi član polja `polje`, tada se članu `polje[i]` može pristupiti pomoću pokazivača `p`, korištenjem izraza `*(p + i)`

Primjer

- Programski zadatak
  - napisati funkciju `najveciClan1D` koja kao rezultat vraća najveću vrijednost u zadanom jednodimenzijskom cjelobrojnom polju
  - napisati glavni program koji će s tipkovnice učitati 10 članova cjelobrojnog polja, pomoću funkcije odrediti najveći član, te ga ispisati na zaslonu
  - primjer izvršavanja programa

```
Upisite clanove > 1 2 3 4 -1 9 -2 9 8 7↵
Najveci clan je 9
```

  - česte pogreške u rješenjima
    - funkcija radi samo s poljima od 10 članova, a treba raditi s poljem koje ima bilo koji broj članova
    - funkcija na zaslon ispisuje rezultat, a rezultat je trebala *vratiti* u funkciju na pozivajućoj razini

Rješenje

```
#include <stdio.h>
#define DIMENZIJA 10

int najveciClan1D(int *p, int n) {
   int najveci, i;

   for (i = 0; i < n; ++i)
      if (i == 0 || *(p + i) > najveci)
         najveci = *(p + i);

   return najveci;
}

int main(void) {
   int polje[DIMENZIJA];
   /* izostavljen je uobicajeni kod za ucitavanje clanova polja */
   printf("Najveci clan je %d", najveciClan1D(polje, DIMENZIJA));

   return 0;
}
```

Ne DIMENZIJA!

Ne 10!

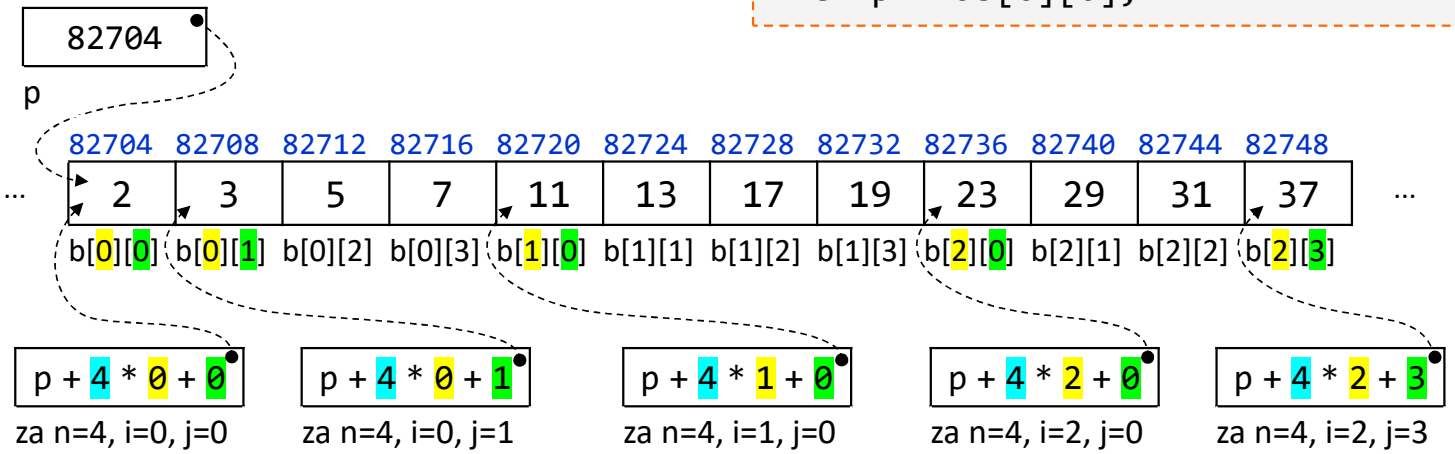
Zašto?

ili &polje[0]

Dvodimenzijska polja i pokazivači

- kako pomoću pokazivača na prvi član polja pristupiti članu polja s indeksom `[i][j]` u polju koje ima `m` redaka i `n` stupaca?

```
int b[3][4] = {{2, 3, 5, 7},
               {11, 13, 17, 19},
               {23, 29, 31, 37}
              };
int *p = &b[0][0];
```



ako je `p` pokazivač na prvi član polja `b` koje ima `n` stupaca, tada se članu polja `b[i][j]` može pristupiti pomoću izraza `*(p + n * i + j)`

Primjer

- Na zaslon ispisati članove nekog dvodimenzijskog polja. Članovima polja pristupati pomoću pokazivača.

```
...
int b[3][4] = {{2, 3, 5, 7},
               {11, 13, 17, 19},
               {23, 29, 31, 37}
              };
int *p = &b[0][0];
int i, j;

for (i = 0; i < 3; ++i) {
   for (j = 0; j < 4; ++j) {
      printf("%5d", *(p + 4 * i + j));
   }
   printf("\n");
}
...
```

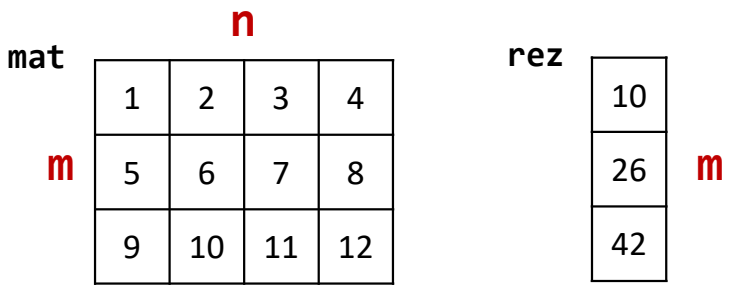
Ime dvodimenzijskog polja kao pokazivač?

- Samo ime **dvodimenzijskog** polja navedeno u nekom izrazu ne može se koristiti kao pokazivač na prvi član tog polja

```
int b[3][4] = {{2, 3, 5, 7},
               {11, 13, 17, 19},
               {23, 29, 31, 37}
              };
int *p = NULL;
p = &b[0][0];  ispravno
p = b[0];      ispravno!
p = b;         neispravno!
```

Primjer

- Programski zadatak
  - Napisati funkciju sumeRedaka koja u zadano jednodimenzijsko polje (članovi tipa int) upisuje sume redaka zadanog dvodimenzijskog polja od m redaka i n stupaca (članovi tipa int).
  - U glavnom programu definirati matricu dimenzije 3 x 4, s tipkovnice učitati članove, pozvati funkciju te na zaslon ispisati dobiveni rezultat.




```
Upisite clanove >↵
1 2 3 4↵
5 6 7 8↵
9 10 11 12↵
Sume redaka su:↵
10↵
26↵
42↵
```

Rješenje

```
#include <stdio.h>
#define BR_RED  3
#define BR_STUP 4

void sumeRedaka(int *mat, int m, int n, int *rez) {
   int i, j;
   for (i = 0; i < m; ++i) {
      *(rez + i) = 0;
      for (j = 0; j < n; ++j)
         *(rez + i) += *(mat + n * i + j);
   }
   return;
}

int main(void) {
   int mat[BR_RED][BR_STUP], rez[BR_RED];
   /* izostavljen je uobicajeni kod za ucitavanje clanova mat */
   sumeRedaka(&mat[0][0], BR_RED, BR_STUP, &rez[0]);
   /* izostavljen je uobicajeni kod za ispis clanova rez*/
   return 0;
}
```

Zašto ovo rješenje nije ispravno?

```
...
int *sumeRedaka(int *mat, int m, int n) {
   int rez[m], i, j;
   for (i = 0; i < m; ++i) {
      rez[i] = 0;
      for (j = 0; j < n; ++j)
         rez[i] += *(mat + n * i + j);
   }
   return &rez[0];
}
...
   int mat[BR_RED][BR_STUP], int *rez, i;
   /* izostavljen je uobicajeni kod za ucitavanje clanova mat */
   rez = sumeRedaka(&mat[0][0], BR_RED, BR_STUP);
   for (i = 0; i < m; ++i)
      printf("%d\n", *(rez + i));
...
```

Primjer

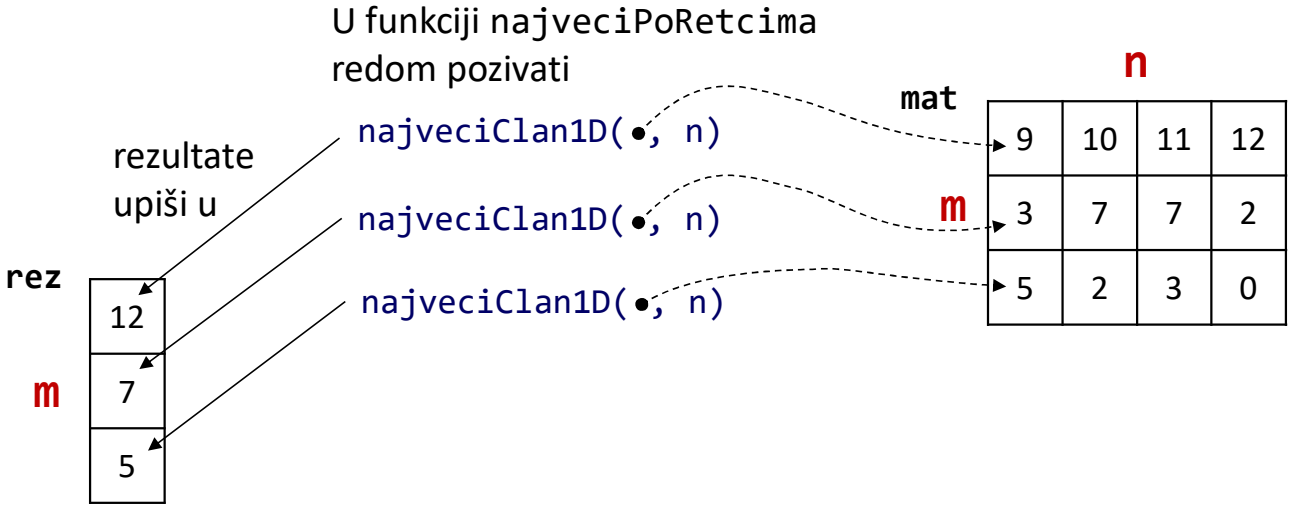
- Programski zadatak
  - Napisati funkciju `najveciPoRetcima` koja u zadanom dvodimenzijskom polju od m redaka i n stupaca (članovi tipa int) pronalazi najveće vrijednosti članova po retcima. Najveće članove po retcima treba vratiti u jednodimenzijskom polju (članovi tipa int)
  - Za traženje najveće vrijednosti u pojedinom retku matrice koristiti već viđenu funkciju `najveciClan1D`

```
int najveciClan1D(int *p, int n) {
   int najveci, i;
   for (i = 0; i < n; ++i)
      if (i == 0 || *(p + i) > najveci)
         najveci = *(p + i);
   return najveci;
}
```

  - U glavnom programu učitati dimenzije matrice m i n, učitati članove matrice, pozvati funkciju te na zaslon ispisati dobiveni rezultat.

Rješenje

- Ideja
  - u funkciji `najveciPoRetcima`, za svaki redak `i` matrice mat, funkciju `najveciClan1D` pozvati s argumentima: pokazivač na prvi član matrice u i-tom retku, duljina retka `n`. Rezultat funkcije upisati na odgovarajuće mjesto (i-ti član) u polju `rez`.



Rješenje (nastavak)

```
#include <stdio.h>

int najveciClan1D(int *p, int n) {
   int najveci, i;
   for (i = 0; i < n; ++i)
      if (i == 0 || *(p + i) > najveci)
         najveci = *(p + i);
   return najveci;
}

void najveciPoRetcima(int *mat, int m, int n, int *rez) {
   int i;
   for (i = 0; i < m; ++i) {
      *(rez + i) = najveciClan1D(mat + n * i + 0, n);
   }
   return;
}
```

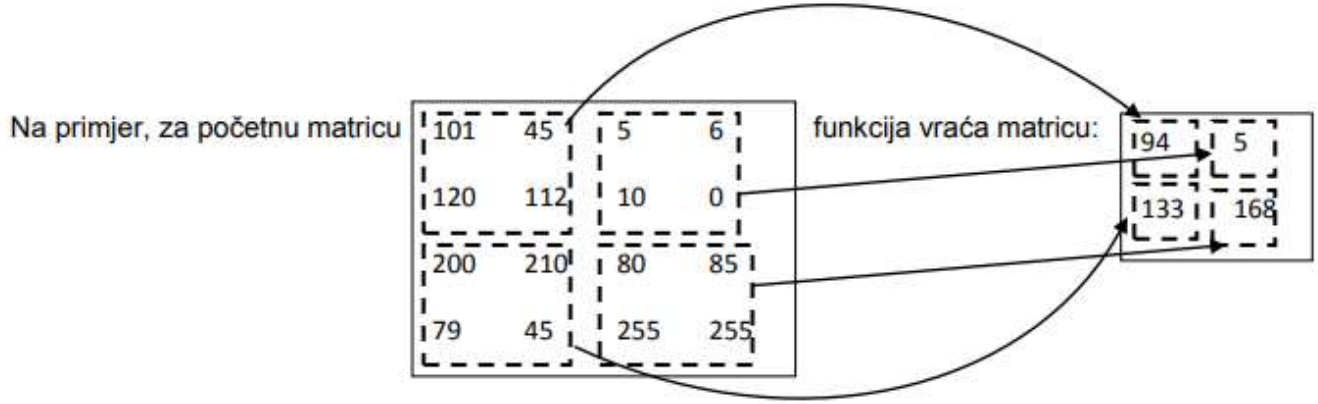
Rješenje (nastavak)

```
...
int main(void) {
   int m, n;  // broj redaka i stupaca matrice mat
   printf("Upisite broj redaka > ");
   scanf("%d", &m);

   printf("Upisite broj stupaca > ");
   scanf("%d", &n);
   int mat[m][n];   // VLA polje!

   /* izostavljen je uobicajeni kod za ucitavanje clanova mat */

   int rez[m];
   najveciPoRetcima(&mat[0][0], m, n, &rez[0]);

   /* izostavljen je uobicajeni kod za ispis rezultata (rez) */
   ...
}
```

Zadatak

Fotografija (dimenzija nxn točaka tj. piksela) je u računalu predstavljena cjelobrojnom matricom dimenzija n x n s vrijednostima u rasponu od 0 do 255 koje predstavljaju odgovarajuću boju (nijansu sive) piksela.

Potrebno je napisati funkciju ***sazmi*** koja će sažeti fotografiju (nxn) za 50% po obje dimenzije stvaranjem nove fotografije (dimenzija (n/2)x(n/2)) u kojoj svaki element nastaje računanjem aritmetičke sredine 4 vrijednosti originalne matrice kao što je prikazano na slici. Aritmetička sredina „zaokružuje“ se na manji cijeli broj



Napisati glavni program u kojem će se učitati dimenzije kvadratne matrice i njeni članovi, pozvati funkcija za sažimanje te ispisati sažeta matrica na zaslon.

Pokazivači i operator *sizeof*

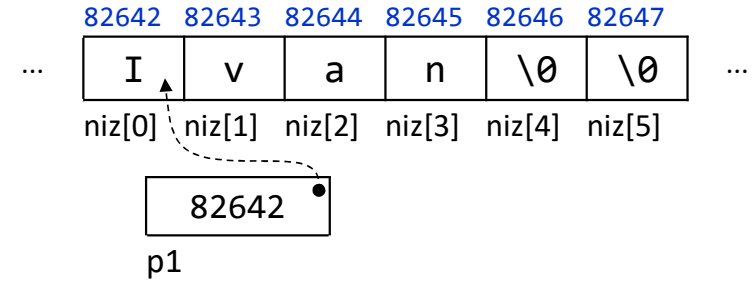
- Voditi računa o tipu objekta nad kojim se primjenjuje operator sizeof

```
void fun(short *p, double broj) {
   printf("%d\n", sizeof(p));            4  (ili 8 na Linuxu)
   printf("%d\n", sizeof(*p));           2
   printf("%d\n", sizeof(broj));         8
   ...
int main(void) {
   short polje[5][10], *p1 = &polje[0][0];
   double x, *p2 = &x;
   printf("%d\n", sizeof(polje));      100
   printf("%d\n", sizeof(p1));           4  (ili 8 na Linuxu)
   printf("%d\n", sizeof(*p1));          2
   printf("%d\n", sizeof(x));            8
   printf("%d\n", sizeof(p2));           4  (ili 8 na Linuxu)
   fun(p1, x);
   ...
```

Pokazivači i nizovi znakova

- za pohranu niza znakova koristi se jednodimenzijsko polje čiji su članovi tipa char, pri čemu se kraj niza obavezno označava članom polja koji sadrži nul-znak '\0'

```
char niz[6] = "Ivan";
char *p1 = &niz[0];
```



Sadržaj *niza znakova* može se mijenjati

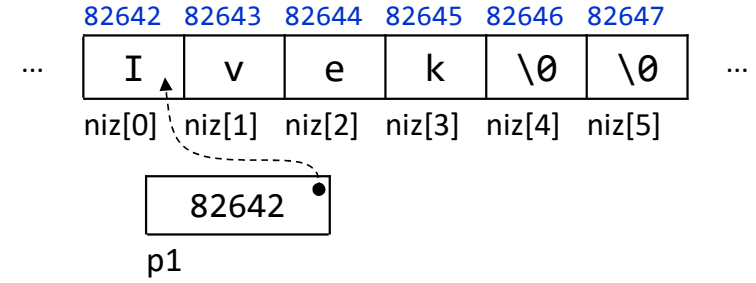
```
*(p1 + 2) = 'e';
*(p1 + 3) = 'k';
```

ili

```
*(niz + 2) = 'e';
*(niz + 3) = 'k';
```

ili

```
niz[2] = 'e';
niz[3] = 'k';
```

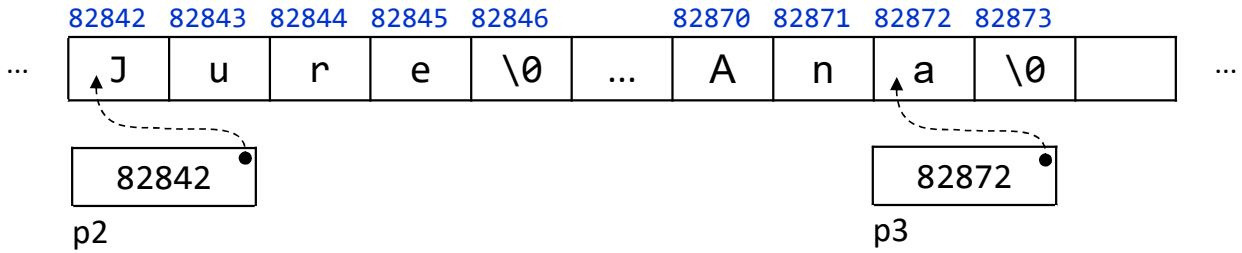


Pokazivači i konstantni znakovni nizovi

- konstantni znakovni niz se u memoriju pohranjuje na jednaki način

```
char *p2 = NULL, *p3 = NULL;
p2 = "Jure";          "Jure" je konstantni znakovni niz
p3 = "Ana" + 2;       "Ana" je konstantni znakovni niz
```

- konstantni znakovni niz se u izrazu evaluira kao pokazivač tipa *pokazivač na char* koji pokazuje na prvi znak u tom konstantnom znakovnom nizu



Sadržaj *konstantnog znakovnog niza* ne može se mijenjati

```
*(p2 + 3) = 'a';          nije dopušteno
*p3 = 'e';                nije dopušteno
printf("%c", *p3);        dopušteno
```

Primjer

- Funkciji `printf` kao argument uz konverzijsku specifikaciju `%s` predaje se pokazivač na prvi znak u nizu kojeg treba ispisati. Što će se ispisati navedenim pozivima funkcije `printf`?

```
char niz[] = "Blaise Pascal";
char *p1 = niz;
printf("%s", niz);                  Blaise Pascal
printf("%s", p1);                   Blaise Pascal
printf("%s", niz + 7);              Pascal
printf("%s", &niz[0] + 7);          Pascal
printf("%s", &niz[7]);              Pascal
printf("%s", p1 + 7);               Pascal

printf("%s", "Isaac Newton");       Isaac Newton
printf("%s", "Isaac Newton" + 4);   c Newton

char *p2 = "Benjamin Franklin";
printf("%s", p2);                   Benjamin Franklin
printf("%s", p2 + 5);               min Franklin
```

Primjer

- Programski zadatak
  - Napisati funkciju u`VelikaSlova` koja kao parametar prima niz znakova i u njemu svako malo slovo zamijeni velikim
  - U glavnom programu jedan niz znakova inicijalizirati na sadržaj "Ivana 123", pozivom funkcije promijeniti niz i promijenjeni niz ispisati na zaslon

Uputa:

Nizovi znakova su jednodimenzijska polja terminirana znakom '\0'.

To znači da se u funkcijama mogu tretirati kao sva ostala jednodimenzijska polja, ali s jednom važnom razlikom: duljinu niza nije potrebno navoditi kao argument jer se duljina niza (ili gdje se nalazi kraj niza) u funkciji može pouzdano utvrditi prema poziciji znaka '\0'.

Rješenje

```
#include <stdio.h>

void uVelikaSlova(char *niz) {
   int i = 0;
   while (*(niz + i) != '\0') {
      if (*(niz + i) >= 'a' && *(niz + i) <= 'z') {
         *(niz + i) = *(niz + i) - ('a' - 'A');
      }
      ++i;
   }
}

int main(void) {
   char ime[] = "Ivana 123";
   uVelikaSlova(ime);
   printf("%s", ime);

   return 0;
}
```

Ova se funkcija ne smije pozvati s argumentom koji je konstantni znakovni niz:

```
uVelikaSlova("Ivana 123");
```

Zašto?

Alternativna rješenja

```
void uVelikaSlova(char *niz) {
   while (*niz != '\0') {
      if (*niz >= 'a' && *niz <= 'z') {
         *niz = *niz - ('a' - 'A');
      }
      ++niz;
   }
}

void uVelikaSlova(char *niz) {
   for (; *niz != '\0'; ++niz) {
      if (*niz >= 'a' && *niz <= 'z') {
         *niz = *niz - ('a' - 'A');
      }
   }
}
```

Primjer

- Programski zadatak
  - Napisati funkciju `nadjiPrviZnak` koja kao parametre prima niz znakova `niz` i znak `z`. Ako znak `z` postoji u nizu, funkcija vraća pokazivač na prvi pronađeni znak `z` u nizu, inače, funkcija treba vratiti *prikladan rezultat* na temelju kojeg će se moći prepoznati da tog znaka u nizu nema
  - U glavnom programu s tipkovnice pročitati niz znakova ne dulji od 50 znakova, pročitati znak `z`, a zatim pomoću funkcije utvrditi gdje se taj znak nalazi te ispisati niz od pronađenog znaka do kraja niza

```
Upisite niz  > Nigdar ni tak bilo da ni nekak bilo↵
Upisite znak > n↵
ni tak bilo da ni nekak bilo
```

```
Upisite niz  > pak ni vezda ne bu da nam nekak ne bu↵
Upisite znak > B↵
Znak B ne postoji u nizu
```

Rješenje

```
#include <stdio.h>

char *nadjiPrviZnak(char *niz, char z) {
   while (*niz != '\0') {
      if (*niz == z)
         return niz;
      ++niz;
   }
   return NULL;
}

int main(void) {
   char niz[50 + 1], z, *rez = NULL;
   printf("Upisite niz  > "); fgets(niz, 50 + 1, stdin);
   printf("Upisite znak > "); scanf("%c", &z);
   rez = nadjiPrviZnak(niz, z);
   if (rez != NULL)
      printf("%s", rez);
   else
      printf("Znak %c ne postoji u nizu", z);
   return 0;
}
```

Primjer

- Programski zadatak
  - Napisati funkciju `nadopuniNiz` koja kao parametre prima nizove znakova `niz1` i `niz2`. Funkcija treba promijeniti `niz1` tako da na njegov kraj dopiše sadržaj niza `niz2`.
  - U glavnom programu prvi niz inicijalizirati na `"Sadrzaj prvog niza"` i ispisati ga na zaslon, drugi niz inicijalizirati na `"Sadrzaj drugog niza"` i ispisati ga na zaslon, pozvati funkciju i na zaslon ispisati novi sadržaj prvog niza

```
Sadrzaj prvog niza↵
Sadrzaj drugog niza↵
Sadrzaj prvog nizaSadrzaj drugog niza
```

Rješenje

```
#include <stdio.h>

void nadopuniNiz(char *niz1, char *niz2) {
   while (*niz1 != '\0')
      ++niz1;
   while (*niz2 != '\0') {
      *niz1 = *niz2;
      ++niz1;
      ++niz2;
   }
   *niz1 = '\0';
}
int main(void) {
   char niz1[37 + 1] = "Sadrzaj prvog niza";
   char niz2[] = "Sadrzaj drugog niza";
   printf("%s\n", niz1);
   printf("%s\n", niz2);
   nadopuniNiz(niz1, niz2);
   printf("%s", niz1);
   return 0;
}
```

Osigurati dovoljno memorije za dopunjavanje niza `niz1`

Zadatak

S tipkovnice učitavati rečenice duljine do 100 znakova te ispisivati najdulju riječ u rečenici i njenu duljinu. Ako u rečenici postoji više jednako dugačkih riječi najveće duljine, ispisati prvu. Učitavanje završiti kada korisnik učita rečenicu bez ijednog znaka (sadrži samo znak ↲).

Najdulju riječ pronaći s pomoću funkcije ***nadjiNajduljuRijec*** koja kao parametre prima:

- pokazivač na znakovni niz (polje) za pohranu rečenice definirano u glavnom programu,
- pokazivač na znakovni niz (polje) definirano u glavnom programu u kojeg treba pohraniti najdulju riječ u rečenici,
- pokazivač na objekt tipa int u koji treba pohraniti duljinu najdulje riječi u rečenici.

Funkcija (i eventualne pomoćne funkcije) ne smiju koristiti pomoćna polja.

Pretpostavite da se u rečenici koriste samo slova engleske abecede, da su riječi razdvojene jednim razmakom i da u rečenici nema interpunkcije niti posebnih znakova (osim jednog znaka novog retka na kraju rečenice koji se nalazi odmah iza zadnje riječi).

Pokazivači

Strukture, funkcije i pokazivači

Struktura jest *modifiable lvalue*

- To znači da se struktura može koristiti kao argument/parametar funkcije, a također se može koristiti i kao rezultat funkcije. Npr.

```
struct tocka_s {double x; double y;};

struct tocka_s
translatiraj(struct tocka_s tocka, double dx, double dy) {
   struct tocka_s novaTocka;
   novaTocka.x = tocka.x + dx;
   novaTocka.y = tocka.y + dy;
   return novaTocka;
}

int main(void) {
   struct tocka_s t1 = {3.0, 4.0}, t2;
   t2 = translatiraj(t1, 2.0, -1.0);
   printf("%lf, %lf => %lf, %lf", t1.x, t1.y, t2.x, t2.y);
```

```
3.000000, 4.000000 => 5.000000, 3.000000
```

Alternativno (pomoću definicije tipa strukture)

```
typedef struct {double x;
                double y;
               } tocka_t ;

tocka_t translatiraj(tocka_t tocka, double dx, double dy) {
   tocka_t novaTocka;
   novaTocka.x = tocka.x + dx;
   novaTocka.y = tocka.y + dy;
   return novaTocka;
}

int main(void) {
   tocka_t t1 = {3.0, 4.0}, t2;
   t2 = translatiraj(t1, 2.0, -1.0);
   printf("%lf, %lf => %lf, %lf", t1.x, t1.y, t2.x, t2.y);
```

Alternativno (iskoristiti parametar)

- Rješenje u kojem funkcija vraća promijenjeni parametar
  - uštedio se prostor na stogu koji se u prethodnom rješenju trošio na lokalnu varijablu novaTocka

```
struct tocka_s {double x; double y;};

struct tocka_s
translatiraj(struct tocka_s tocka, double dx, double dy) {
   tocka.x += dx;
   tocka.y += dy;
   return tocka;
}
```

- U oba rješenja argument (varijabla `t1`) se nije promijenio
  - međutim, što ako položaj točke kakav je bio prije translacije nije potrebno pamtiti? Npr.

```
t1 = translatiraj(t1, 2.0, -1.0);
```

Pokazivač na strukturu

- Funkcija kao parametar može koristiti pokazivač na strukturu

```
struct tocka_s {double x; double y;};
void translatiraj(struct tocka_s *pTocka, double dx, double dy) {
   (*pTocka).x += dx;
   (*pTocka).y += dy;
   return;
}

int main(void) {
   struct tocka_s t1 = {3.0, 4.0};
   printf("%lf, %lf", t1.x, t1.y);
   translatiraj(&t1, 2.0, -1.0);
   printf(" => %lf, %lf", t1.x, t1.y);
```

- umjesto cijele strukture, na stog se u ovom slučaju kopira samo pokazivač na strukturu, a funkcija pomoću dobivenog pokazivača mijenja sadržaj varijable `t1`

Alternativno (pomoću definicije tipa strukture)

```
typedef struct {double x;
                double y;
               } tocka_t;

void translatiraj(tocka_t *pTocka, double dx, double dy) {
   (*pTocka).x += dx;
   (*pTocka).y += dy;
   return;
}

int main(void) {
   tocka_t t1 = {3.0, 4.0};
   printf("%lf, %lf", t1.x, t1.y);
   translatiraj(&t1, 2.0, -1.0);
   printf(" => %lf, %lf", t1.x, t1.y);
```

Operator pristupa članu strukture preko pokazivača

- Binarni operator `->`. U literaturi također: strelica, *arrow operator*
  - lijevi operand je pokazivač na strukturu, desni operand je ime člana strukture, rezultat operacije je član strukture

```
struct osoba_s {
  char prezime[40+1];
  char ime[40+1];
  int visina;
};
struct osoba_s osoba, *pOsoba = &osoba;
scanf("%s", (*pOsoba).prezime);      ili pOsoba->prezime
scanf("%s", (*pOsoba).ime);          ili pOsoba->ime
scanf("%d", &(*pOsoba).visina);      ili &pOsoba->visina
...
printf("%s", (*pOsoba).prezime);     ili pOsoba->prezime
printf("%s", (*pOsoba).ime);         ili pOsoba->ime
printf("%d", (*pOsoba).visina);      ili pOsoba->visina
```

Član strukture može biti polje

- I struktura koja sadrži polje smije se koristiti kao parametar
  - Primjer: podaci o studentu i bodovima stečenim na predmetu UPRO pohranjeni su u sljedećoj strukturi

```
struct bodovi_s {
   char jmbag[13 + 1];
   char ime[50 + 1];
   char prezime[50 + 1];
   float mi;
   float zi;
   float lab[8];
};
```

Ukupno 156 bajtova

  - napisati funkciju koja kao parametar prihvaća prikazanu strukturu, a kao rezultat vraća cijeli broj koji predstavlja odgovarajuću ocjenu
    - korištenje strukture koja sadrži polje kao parametar funkcije nije zabranjeno, ali treba voditi računa o utrošku memorije (stog)

Rješenje

```
#include <stdio.h>
struct bodovi_s {
   char jmbag[13 + 1]; ...
};
int izracunajOcjenu(struct bodovi_s bodovi) {
   int suma = bodovi.mi + bodovi.zi + bodovi.lab[...] ...
   if (suma >= 50.0f && suma < 62.5f) ocjena = 2; else ...
   return ocjena;
}
int main(void) {
   struct bodovi_s bodoviHorvat;
   ...
   printf("Ocjena je %d", izracunajOcjenu(bodoviHorvat));
   return 0;
}
```

Alternativno rješenje koje *štedi* stog

```
#include <stdio.h>
struct bodovi_s {
   char jmbag[13 + 1]; ...
};
int izracunajOcjenu(struct bodovi_s *bodovi) {
   float suma = bodovi->mi + bodovi->zi + bodovi->lab[...] ...
   if (suma >= 50.0f && suma < 62.5f) ocjena = 2; else ...
   return ocjena;
}
int main(void) {
   struct bodovi_s bodoviHorvat;
   ...
   printf("Ocjena je %d", izracunajOcjenu(&bodoviHorvat));
   return 0;
}
```

Ne zloupotrebljavati to svojstvo struktura!

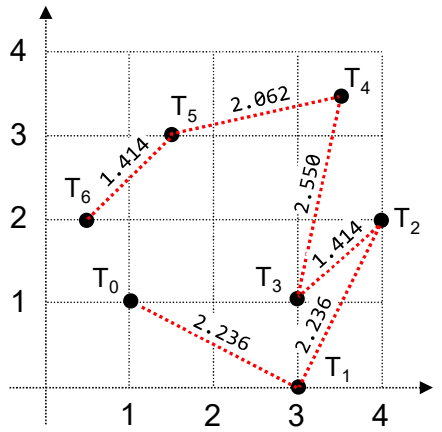
- U vrlo lošim programima ta se mogućnost zloupotrebljava tako što se "polje omota strukturom" s jedinim ciljem da ga se "prenese u funkciju". To je **vrlo loša** ideja i tako se **ne smije** raditi.

```
struct omotanoPolje_s {
   int polje[1000];
};

int najveciClan1D(struct omotanoPolje_s omotanoPolje, int n) {
   ...
   return najveci;
}

int main(void) {
   struct omotanoPolje_s omotanoPolje;
   ...
   printf("Najveci je %d", najveciClan1D(omotanoPolje, n));
```

Za koliko bajtova se poveća sadržaj stoga kada se pozove funkcija?

Primjer

- Programski zadatak
  - Linija je u Kartezijevom koordinatnom sustavu opisana nizom točaka. Za pohranu podataka o jednoj točki (koordinate x i y, tipa `double`) koristi se struktura `tocka_s`. Podaci o točkama koje predstavljaju liniju pohranjeni su u jednodimenzijskom polju čiji su članovi strukture `tocka_s`.




```
struct tocka_s {double x;
                double y;
               };
```

```
struct tocka_s linija[n];
```

x	1.0	x	3.0	x	4.0	x	3.0	x	3.5	x	1.5	x	0.5
y	1.0	y	0.0	y	2.0	y	1.0	y	3.5	y	3.0	y	2.0

Primjer

- Programski zadatak (nastavak)
  - Napisati funkciju koja izračunava ukupnu duljinu linije koja je predstavljena jednodimenzijskim poljem čiji su članovi strukture `tocka_s`. U glavnom programu učitati broj i koordinate točaka, pozivom funkcije izračunati, a zatim na zaslon ispisati duljinu linije
- Primjer izvršavanja programa

```
Upisite broj tocaka linije > 7↵
Upisite koordinate tocke T0 > 1.0 1.0↵
Upisite koordinate tocke T1 > 3.0 0.0↵
Upisite koordinate tocke T2 > 4.0 2.0↵
Upisite koordinate tocke T3 > 3.0 1.0↵
Upisite koordinate tocke T4 > 3.5 3.5↵
Upisite koordinate tocke T5 > 1.5 3.0↵
Upisite koordinate tocke T6 > 0.5 2.0↵
Ukupna duljina linije je    11.912 (jed.mj.)
```

Rješenje (1. dio)

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

struct tocka_s {double x;
                double y;
               };

/* izracunava udaljenost medju tockama t1 i t2 */
double udaljenost(struct tocka_s *t1, struct tocka_s *t2) {
   return sqrt(pow(t2->x - t1->x, 2.) + pow(t2->y - t1->y, 2.));
}

/* izracunava duljinu linije odredjene tockama u polju linija */
double duljinaLinije(struct tocka_s *linija, int n) {
   double duljina = 0.;
   for (int i = 1; i < n; ++i) {
      duljina += udaljenost(linija + i, linija + i - 1);
   }
   return duljina;
}
```

Rješenje (2. dio)

```
int main(void) {
   int n;
   printf("Upisite broj tocaka linije > ");
   scanf("%d", &n);

   struct tocka_s linija[n];

   /* ucitaj koordinate za n tocaka */
   for (int i = 0; i < n; ++i) {
      printf("Upisite koordinate tocke T%d > ", i + 1);
      scanf("%lf %lf", &linija[i].x, &linija[i].y);
   }

   printf("Ukupna duljina linije je %9.3f (jed.mj.)"
         , duljinaLinije(linija, n));

   return 0;
}
```

Prije sljedećeg predavanja

- Edgar:
  - Tutorial: **nema**
  - **16. vježbe uz predavanja**