acs
applied computer science

UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Osnovi programiranja i programskih jezika

Računarske vežbe – vežba 6

Zimski semestar 2025/2026.

Studijski program: Informacioni inženjering

Pokazivači
Dinamička alokacija memorije

Pokazivači

Pokazivač je promenljiva koja sadrži adresu u memoriji, tj. broj

- ukazuje na lokaciju u memoriji
- sadržaj pokazivačke promenljive je adresa (lokacija u memoriji)

Pokazivačke promenljive pokazuju na druge promenljive ili na početak memorijskog bloka

- pokazivači na promenljive tipa int, float, itd.
- generički pokazivač tipa void*

Pokazivači

Deklaracija:

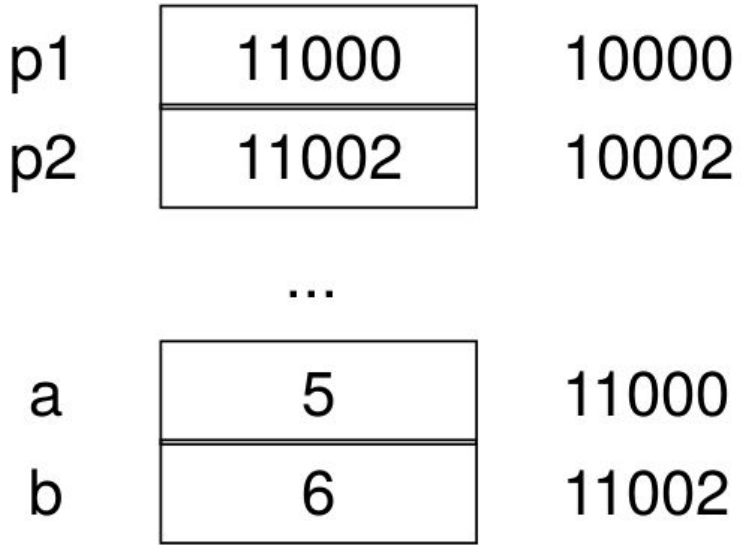
```
int a = 5;
int b, c;
int *p1, *p2;
```

Dodela vrednosti:

```
p1 = &a;
p2 = &b;
```

Pristup lokaciji:

```
c = *p1; // c <- a
*p2 = 6; // b <- 6
```



Operator referenciranja

- Unarni operator referenciranja & daje adresu promenljive
- Izraz p = &a dodeljuje adresu promenljive **a** promenljivoj **p**, dakle onda **p** pokazuje na **a**
- Da bi se prikazala vrednost pokazivača koristi se konverzija %p

Operator dereferenciranja

Primenom unarnog operatora dereferenciranja * može se posredno pristupiti nekom podatku pomoću memorijske adrese

Specijalna konstanta NULL

- Konstanta koja se nalazi u `stdio.h`
- Ako pokazivačka promenljiva ima vrednost `NULL`, onda ne pokazuje ni na šta
- Primer:

```
int *p;
p = NULL;
```

- Neinicijalizovana vrednost NIJE `NULL`!
  - mora se eksplicitno inicijalizovati na `NULL`

Primer I

```c
#include <stdio.h>

int main() {
  int i;
  int *pi;

  i = 7;
  pi = &i;

  printf("Promenljiva - adresa:\t %p, vrednost:\t %d\n\n", &i, i);
  printf("Pokazivac - adresa:\t %p, vrednost:\t %p\n\n", &pi, pi);
  printf("Pokazivac - vrednost:\t %p, sadrzaj:\t %d\n\n", pi, *pi);

  i = 10;

  printf("Pokazivac - vrednost:\t %p, sadrzaj:\t %d\n\n", pi, *pi);

  (*pi)++;

  printf("Promenljiva - adresa:\t %p, vrednost:\t %d\n\n", &i, i);

  return 0;
}
```

Adresna aritmetika

Osim adresnih, nad pokazivačima su moguće i sledeće operacije:

- sabiranje pokazivača i broja
- oduzimanje broja od pokazivača
- oduzimanje dva pokazivača
- poređenje dva pokazivača

Pokazivači i nizovi

- Identifikator (ime) niza je zapravo pokazivač na prvi element u memoriji (koja je dodeljena tom nizu)
- Primer:

```
int a[20];  // a – je identifikator niza
int *pa;

pa = a;   ⇔   pa = &a[0];

a+2 ⇔ &a[2] ⇔ pa+2

*(a+3) ⇔ a[3] ⇔ *(pa+3)
```

Primer 2 - pristup elementima niza preko pokazivača.

```c
#include <stdio.h>

int main() {
  int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
  int *pa;
  int i;

  pa = a;
  printf("%d\n\n", *pa);

  printf("%d\n\n", *(a+1));

  // ispis elemenata sa parnim indeksom
  for (i=0; i<10; i+=2) {
    pa = a+i;
    printf("%d\n", *pa);
  }

  return 0;
}
```

Vrlo česte greške

- Nemoguće je definisati pokazivač na konstantu ili izraz
- Nemoguće je promeniti adresu promenljive (jer to i ne zavisi od nas)
- Zbog prethodnog sledeći iskazi su pogrešni:
  - `i = &3;`
  - `j = &(k+5);`
  - `k = &(a==b);`
  - `&a = &b;`
  - `&a = 150;`

Zadatak 1.

Nacrtati dijagram toka algoritma i napisati C program kojim se učitava niz celobrojnih elemenata (maksimalno 30) i zatim obrće redosled elemenata (zamena prvog sa poslednjim, drugog sa pretposlednjim, itd.) unesenog niza. Za pristup elementima niza koristiti pokazivače.

Dinamička alokacija memorije

- Zauzimanjem memorije za maksimalan broj članova niza (primer: `int a[MAX]`) nameće se pitanje može li se, u programskom jeziku C, zauzeti tačno onoliko memorije koliko je potrebno za n elemenata, gde n unosi korisnik
- Problem je što je dužina niza poznata tek u vreme izvršavanja, a ne u vreme prevođenja programa
- Dinamička alokacija memorije omogućuje zauzimanje proizvoljne količine memorije u toku izvršavanja programa

Dinamička alokacija memorije

- Sav dinamički zauzeti memorijski prostor potrebno je eksplicitno osloboditi po prestanku korišćenja
- U programskom jeziku C teret je prebačen na programera da upravlja dinamički zauzetom memorijom!

Funkcije za dinamičko upravljanje memorijom

```
void *malloc(size_t size);
void *calloc(int n, size_t size);
void *realloc(void *ptr, size_t size);
void free(void *ptr);
```

Primer 3

```c
#include <stdio.h>
#include <stdlib.h> // definicije f-ja malloc i free

int main () {
    int *br;
    br = malloc (sizeof(int));

    printf("Unesite celi broj: ");
    *br = 5; // scanf("%d", br);

    printf("Ucitan broj je: %d\n\n", *br);
    free(br); // oslobadjanje (dealociranje)

    return 0;
}
```

Zadatak 2.

Modifikovati Zadatak 1 tako da koristi dinamičku alokaciju memorije.

Zadatak 3.

Nacrtati dijagram toka algoritma i napisati C program kojim se unosi niz struktura (proizvoljne dužine) tipa Tačka sa poljima: x koordinata i y koordinata. Na standardni izlaz ispisati koordinate centroida (težišta) unesenog niza tačaka.

Zadatak 4.

Nacrtati dijagram toka algoritma i napisati C program kojim se unose dimenzije matrice A, a potom i njeni elementi. U unesenoj matrici odrediti i ispisati najveći element koji se nalazi na obodu matrice.

Zadatak 5.

Nacrtati dijagram toka algoritma i napisati C program kojim se unosi niz struktura tipa **Student** sa poljima: ime, prezime, broj položenih ispita i niz ocena sa tih ispita. Potom se računa i ispisuje za svakog studenta: ime, prezime i prosečna ocena. Za niz struktura i niz ocena unutar strukture koristiti dinamičku alokaciju memorije.

Zadatak za domaći 1.

Nacrtati dijagram toka algoritma i napisati C program kojim se unosi niz struktura tipa **Zaposleni** sa poljima: ime, prezime, JMBG, koeficijent stručne spreme i varijabilni koeficijent. Plata zaposlenog se računa po formuli:

plata = (koef. stručne spreme + var. koef) * cena rada

Cenu rada unosi korisnik. Potom se računa i ispisuje za svakog zaposlenog ime, prezime i plata za tekući mesec. Za niz struktura koristiti dinamičku alokaciju memorije.

Zadatak 6.

Nacrtati dijagram toka algoritma i napisati C program: Korisnik unosi brojeve sve dok ne unese 0. Brojeve treba čuvati u nizu za koji treba da se alocira prostor prvo za jedan element a potom se vrši realokacija za 2 puta veći prostor (1 > 2 > 4 > 8 > 16 itd.). Ispisati sve unešene članove niza.

Zadatak za domaći 2.

Za unešeni niz iz zadatka 6, naći sve jedinstvene brojeve. Za te brojeve treba alocirati niz struktura **Histogram** koja imaju polja {*broj*, *broj_pojavljivanja*}. Ova polja čuvaju po jedan broj koji se pojavljivao u nizu i broj pojavljivanja tog broja. Treba ispisati sadržaj novog niza i srednju vrednost kao: SUM(*broj***broj_pojavljivanja*)/*broj_svih_brojeva*.

Zadatak za domaći 3.

Nacrtati dijagram toka algoritma i napisati C program: Treba voditi evidenciju koja mesta u pozorištu su zauzeta. Na početku se unose broj redova i broj stolica po redu. Treba alocirati *short* za svako sedište. Ukoliko je slobodno treba da stoji 0 za to sedište, a 1 ako je zauzeto. Na početku su sva mesta slobodna. Evidencija zauzetih mesta se vrši tako što se unosi lokacija sedišta kao par vrednosti (broj_reda, broj_mesta_u_redu). Rad se završava kada se za bar jedan broj u paru unese broj manji od 1. Nakon prekida unosa, treba ispisati sva sedišta koja su zauzeta i ispisati broj uzastopno zauzetih mesta.