Programski jezici i strukture podataka

5

POKAZIVAČI

Pokazivači

- Referenca na podatak tipa objekta ili na funkciju.
- Primene:
  - pri definisanju funkcija koje se pozivaju po referenci
  - pravljenje dinamičkih struktura
    - spregnute liste
    - stabla...

Pokazivači

- Pokazivač je adresa u memoriji, broj.
  - ukazuje na lokaciju u memoriji i tip objekta ili funkcije
  - 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.

Adrese u memoriji

- Operativna memorija je niz memorijskih lokacija koje su numerisane celim brojevima 0, 1, 2,*m*-1, pri čemu je *m* kapacitet memorije.
- Brojevi pridruženi memorijskim lokacijama nazivaju se ***adresama.***
- Najmanja memorijska lokacija koja može samostalno da se adresira je obično jedan bajt koji se sastoji od 8 bitova.
- Podaci različitih tipova smeštaju se u jedan ili više uzastopnih bajtova.
- Na primer char u 1 bajt, short int u 2 bajta, long int i float u 4 bajta, a double u 8 bajtova.

Pokazivač

- ***Pokazivač*** je prost podatak u koji može da se smesti adresa neke lokacije u memoriji. Pokazivači obično zauzimaju 4 bajta.
- Broj bajtova koje zauzima pokazivač zavisi od mogućeg opsega adresa na datom računaru, a ne od broja bajtova koliko zauzimaju pokazivani podaci.
- Kada neki podatak zauzima više bajtova, pod adresom podatka podrazumeva se bajt sa najmanjim rednim brojem.

Deklarisanje pokazivača

- Sintaksa deklarisanja pokazivača na objekat koji nije niz:

  type * [*lista-kvalifikatora-tipa*] *ime*[= *inicijalizator*];

- U deklaracijama * znači pokazivač na.
- Identifikator *ime* ***deklariše se kao objekat tipa*** *tip* ****, ili kao pokazivač na tip*** *tip*.

Definisanje pokazivača

- Opis tipa na početku deklarativne naredbe označava tip pokazivanog podatka.
- Pokazivači se definišu, kao i svi ostali podaci, uobičajenim naredbama za definisanje , s tim što se u nazivu podatka dodaje modifikator * ispred identifikatora podatka:

```
* identifikator_pokazivaca
```

  - identifikator standardnih prostih tipova,
  - identifikator tipa ranije uveden naredbom typedef
  - opis tipa koji definiše programer (na primer opis nabrajanja – enum)

Referenciranje

- Unarni operator & daje adresu promenljive
- Izraz p=&a dodeljuje adresu promenljive a promenljivoj p, pa sada p pokazuje na a
- Da bi se odštampala vrednost pokazivača koristi se konverzija %p.

Referenciranje

- Adresa nekog podatka u memoriji može da se dobije pomoću prefiksnog unarnog operatora &.
- Operand operatora & mora da bude podatak koji se nalazi u memoriji.
- Operator & **ne sme** da se primeni na prolazne podatke za koje je prilikom definisanja traženo da se smeste u procesorske registre (modifikator tipa register)

Dereferenciranje

- Primenom unarnog operatora * može se posredno pristupiti nekom podatku pomoću memorijske adrese.
- Dohvatanje podatka posredno, pomoću adrese, naziva se i ***indirektnim adresiranjem.***
- Zato se i sam operator * ponekad naziva operatorom indirektnog adresiranja.

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 NEMA NULL!
  - mora se eksplicitno inicijalizovati na NULL

Pokazivači na vrednost NULL

- Pokazivač na vrednost `NULL` (null pointer) dobija se kada se konvertuje ***konstantan pokazivač*** na vrednost `NULL` u tip pokazivač (**`z19.c`**).
- ***Konstanta pokazivača*** na vrednost `NULL` jeste celobrojan konstantni izraz vrednosti `0`, konvertovan na tip `void *`.

Pokazivač na tip void

- Pokazivač na tip void predstavlja pokazivač tipa void *.
- Objekti tipa void ne postoje, zato se pokazivač ovog tipa koristi kao pokazivač opšte namene.
- On može da predstavlja adresu bilo kog objekta, ali ne i njegov tip.
- Da bipristupili objektu u memoriji ovakav pokazivač se uvek kastuje na odgovarajući tip.

Notacija

- Notacija int *p može da se tumači ovako: ako je *p podatak tipa int, p mora da je pokazivač na takve podatke.
- Treba posebno naglasiti da u slučaju definisanja više pokazivača jednom naredbom, modifikator * stavlja se ispred svakog identifikatora.

Inicijalizacija pokazivača

- Pokazivačke promenljive nakon deklaracije imaju nedefinisanu početnu vrednost.
- Pokazivač se može inicijalizovati pomoću sledećih vrsta inicijalizacije:
  - konstantan pokazivač na vrednost NULL
  - Pokazivač na isti tip
  - Pokazivač na tip void
- Inicijalizator pri definisanju pokazivača treba da bude adresa u operativnoj memoriji. Na primer, naredbom:

```
dotible x, *px=&x;
```

- definiše se podatak x tipa double i pokazivač px na podatke tipa double.
- Pokazivač px se inicijalizuje adresom (&x) prethodno definisane promenljive x.

Nepromenljivi i nepostojani podaci i pokazivači

- Do sad definisani pokazivači su ***promenljivi pokazivači na promenljive podatke.***
- To znači da vrednost takvih pokazivača može da se promeni, a takođe, pomoću takvih pokazivača može da se promeni vrednost pokazivanog podatka.
- Ako se prilikom definisanja pokazivača, na početku naredbe za definisanje podataka dodaje modifikator **const** dobiće se ***promenljivi pokazivač na nepromenljivepodatke.***
- Nepromenljivost se odnosi na pokazivane podatke, a ne na pokazivače same.

```
const tip *p;
```

- p je pokazivač na nepromenljive (const) podatke tipa *tip.*
- Vrednost pokazivača na nepromenljive podatke može da se promeni, ali pomoću njega ne može da se promeni vrednost podatka na koji on pokazuje.

Nepromenljivi i nepostojani podaci i pokazivači

- Nepromenljivi pokazivači se dobijaju korišćenjem modifikatora ***const** ispred samog identifikatora definisanog pokazivača.
- Vrednosti takvih pokazivača ne mogu da se promene kasnije u programu, pa moraju da se navedu i inicijalizatori.

```
tip *const p = &k;
```

- Nepromenljivi pokazivač na promenljive podatke.
- p je nepromenljivi (const) pokazivač (*) na podatke tipa tip.

Nepromenljivi i nepostojani podaci i pokazivači

```
const tip *const p = &k;
```

- ***nepromenljivi pokazivač na nepromenljive podatke.***
- p je nepromenljivi (const) pokazivač (*) na nepromenljve (const) podatke tipa *tip*.

Adresna aritmetika

- U jeziku C dozvoljene su sledeće operacije nad pokazivačima:
  - dodela vrednosti jednog pokazivača drugom,
  - dodavanje celobrojnog podatka na vrednost pokazivača i oduzimanje celobrojnog podatka od vrednosti pokazivača,
  - oduzimanje i upoređivanje dva pokazivača,
  - upoređivanje pokazivača nulom.

Adresna aritmetika -
dodela vrednosti jednog pokazivača drugom

- Operatorom **=** može da se dodeljuje vrednost pokazivača datog tipa drugom pokazivaču istog tipa.
- Ako ta dva pokazivača pokazuju na podatke različitih tipova, obavezna je upotreba operatora za konverziju tipa *(cast* operatora).
- Konverzija tipa nije obavezna ako je jedan od pokazivača generički pokazivač (tip **void** *).

Adresna aritmetika
dodavanje i oduzimanje celobrojnog podatka

- Operatorima +, ++ i += može da se izračuna zbir vrednosti pokazivača i celobrojnog podatka, a operatorima -, -- i -= da se izračuna razlika vrednosti pokazivača i celobrojnog podatka.
- Pokazivač ne sme da bude tipa **void** * (ne zna se šta je jedinica mere).
- Jedinica mere pri sabiranju i oduzimanju vrednosti pokazivača i celog broja je veličina pokazivanih podataka.
- Na taj način, ako pokazivač **p** pokazuje na neki element niza, **p+1** pokazuje na naredni a **p-1** na prethodni element, bez obzira na tip elemenata niza.
- Posledice su nepredvidljive ako rezultat aritmetičke operacije pokazuje na neku lokaciju izvan niza na koga pokazuje pokazivač **p**.

Adresna aritmetika
oduzimanje i upoređivanje dva pokazivača

- Dozvoljeno je oduzimanje operatorom **-** i upoređivanje relacijskim operatorima <, <=, > i >= dva pokazivača istog tipa (ali različitog od **void** *) koji pokazuju na elemente istog niza.
- Rezultati tih operacija su isti kao da su operacije primenjivane na indekse unutar niza.
- Nema smisla oduzimanje i upoređivanje pokazivača koji pokazuju na međusobno nezavisne podatke, jer ne postoji uvid u redosled smeštanja podataka u memoriju.
- Dozvoljeno je upoređivanje relacijskim operatorima ==, !=, dva pokazivača istog tipa (uključujući i **void** *). Time se dobija odgovor da li oba pokazivača pokazuju na isti podatak ili ne.

Adresna aritmetika
upoređivanje pokazivača nulom

- Dodzvoljeno je upoređivanje pokazivača proizvoljnog tipa (uključujući i **void** *) sa nulom.
- Time se ispituje da li pokazivač uopšte pokazuje na neki podatak.
- Vrednost nula označava da pokazivač ne pokazuje ni na koji podatak.
- Jasnoće radi, preporučuje se za te svrhe korišćenje simboličke konstante NULL koja je definisana u standardnim zaglavljima <stdio.h> i <stdlib.h>.

Pokazivači i nizovi

- Nizovi se mogu posmatrati kao pokazivači
- Kada se definiše niz, alocira se navedeni broj memorijskih lokacija za smeštaj elemenata niza. Promenljiva koja predstavlja niz se postavlja tako da pokazuje na prvu od ovih lokacija.

Pokazivači i nizovi

- Identifikator 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;     je isto što i:    pa = &a[0];

a+2    == &a[2] == pa+2

*(a+3) == a[3]  == *(pa+3)
(z22.c)
```

Pokazivači i nizovi

- Može se smatrati da je identifikator niza **nepromenljivi pokazivač** na podatke čiji je tip jednak tipu elemenata niza.
- Ako je niz definisam sa **int a [10],** može se smatrati da je identifikator **a** definisan sa **int *const a** i inicijalizovan početnom adresom bloka memorije veličine 10 podataka tipa **int.**
- Na osnovu definicije sabiranja pokazivača i celih brojeva, važe sledeće ekvivalencije izraza adresne aritmetike i indeksiranja:
- **&a[i]** isto što i: **a+i** isto što i: **i+a**
- **a[i]** isto što i: ***(a+i)** isto što i: ***(i+a)** isto što i: **i[a]**

Smeštanje višedimenzionalnšh nizova u memoriju

- Višedimenzionalni su nizovi čiji su elementi takođe nizovi.
- U slučaju matrica (dvodimenzionalni nizovi) svaka vrsta je jedan vektor (jednodimenzionalni niz), odnosno matrica je niz vektora.
- Trodimenzionalni niz, po analogiji, je niz matrica.
- Za slučaj matrica, elementi se smeštaju po vrstama (prva vrsta na početku, iza nje druga, pa treća, itd.). Kod trodimenzionalnog niza se na sličan načinređaju komponentne matrice jedna za drugom.

Smeštanje višedimenzionalnšh nizova u memoriju

- Indeksiranje za pristup elementima može da se zameni odgovarajućim izrazima adresne aritmetike:

a[i][j] je isto kao i: ***((tip*)a + n*i +j)**

- Za više dimenzionalne nizove formula je analogna ovoj formuli.

Ograničeni pokazivači

- Pokazivač sa kvalifikatorom restrict zove se ograničeni pokazivač.
- Između ograničenog pokazivača i objekta na koji pokazuje postoji sledeća veza:
  - tokom životnog veka pokazivača, objekat se može menjati ili mu se **pristupati samo pomoću tog pokazivača**.

```
typedef struct { long kljuc; // Definiše tip strukture.
                 /* ... ostali članovi... */
                   } Data_t;
Data_t * restrict rPtr = malloc( sizeof(Data_t) );
// Zauzima memoriju za strukturu.
```

Pokazivači i funkcije

```
#include <stdio.h>
// PO VREDNOSTI
void f(int i) {
    i = 3;
}

int main() {
    int i = 5;
    f(i);
    printf("%i", i);
    return 0;
}
```

```
#include <stdio.h>
// PO REFERENCI
void f(int *i) {
    *i = 3;
}

int main() {
    int i = 5;
    f(&i);
    printf("%i", i);
    return 0;
}
```