Programski jezici i strukture podataka

5

Adresna aritmetika

- 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.

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** *).

Dodavanje i oduzimanje celobrojnog podatka

- Operatorima +, ++ i += može da se izračuna zbir vrednosti pokazivača i celobrojnog podatka
- 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

- 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**.

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.
- 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.

Upoređivanje pokazivača nulom

- Dodzvoljeno je upoređivanje pokazivača proizvoljnog tipa (uključujući i **void** *) sa nulom (NULL).
- Time se ispituje da li pokazivač uopšte pokazuje na neki podatak.

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]**

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;
}
```

FUNKCIJE

Definicije funkcija

- Sve instrukcije se nalaze unutar funkcija.
- Funkcijom main() započinje izvršavanje programa.
- Funkcija se definiše jednom, deklariše i poziva proizvoljan broj puta.

Blok funkcije

Specifikatori memorijskih oblasti (*storage class specifier)*

- `extern` (podrazumevana vrednost)
- `static` (ne vidi se u ostalim izvornim datotekama - skrivena je)
- `register` (namenjen deklaraciji parametara - govori prevodiocu da promenljivu učini što pre dostupnom)
- `inline` (uputstvo prevodiocu da na mesto funkcije direktno umetne mašinski kod)

Parametri funkcije su

- Lokalne promenljive
- Oblast važenja im je funkcija

Niz kao parametar funkcije

- Ako funkciji treba poslati niz potrebno je deklarisati odgovarajući parametar:
  `tip ime[]`
- Imena nizova se automatski konvertuju u pokazivače pa je ekvivalent:
  `tip *ime`
- Ako se prilikom deklarisanja parametra koristi konstanta:
  `tip ime[30]` - zanemaruje se.

funkcija `main()`

- Okruženje za samostalno izvršavanje programa (program se izvršava bez podrške operativnog sistema - sveden na minimalne mogućnosti standardne biblioteke) - ime i tip funkcije koja će se izvršiti prva zavisi od implementacije.
- Okruženje koje obezbeđuje podršku izvršavanju programa (program se izvršava pod kontrolomi uz podršku operativnog sistema) - izvršavanje počinje od `main()`

Program preveden za sistem koji obezbeđuje podršku izvršavanju programa mora definisati funkciju `main()`

- Funkcija bez parametara sa rezultatom tipa `int`.

```
int main(void)
```

- Funkcija sa dva parametratipa `int` i `char **` i sa rezultatom tipa `int`. (**z23.c**)

```
int main(int argC, char *argV[])
```

- Funkcija sa tri parametratipa `int` i dva `char **`, i sa rezultatom tipa `int`.

```
int main(int argC, char *argV[], char *envP[])
```

Deklaracije funkcija (prototip)

- Deklaracija opisuje interfejs funkcije:

```
extern int preimenuj(const char
*, const char *);
```

Pokazivači kao argumenti i rezultati funkcije

- Poziv po vrednosti (prirodan C jeziku, kopira vrednosti argumenata u parametre funkcije - lokalne promenljive za funkciju)
- Poziv po referenci (funkcija može direktno da pristupi bilo kojoj promenljivoj vidljivoj pozivaocu, ako ima njenu adresu kao argument)(**z24.c**)

Umetnute funkcije (`inline`)

- Česta upotreba malih funkcija usporava izvršavanje.
- Uputstvo prevodiocu da na mesto takvih funkcija direktno umetne mašinski kod olakšava izvršavanje. (**z25.c**).
- Funkcije koje sadrže cikluse, obično nije preporučljivo definisati kao umetnute.

Rekurzivne funkcije

- Funkcija koja poziva samu sebe (direktno ili indirektno).
- Mora postojati uslov za prekid izvršavanja.
- Primer binarno pretraživanje (**z26.c**).