Programiranje i programski jezici

Strukturne promenljive i pokazivači

Strukturne promenljive imaju adrese pa je moguće definisati pokazivačke promenljive koje će pokazivati na njih. Da bi se definisao pokazivač na strukturnu promenljivu treba deklarisati pokazivačku promenljivu na objekat istog tipa kao što je tip strukturne promenljive i dodeliti mu adresu strukturne promenljive.

```
struct licnost *osoba
```

Elementima strukture na koju ova promenljiva pokazuje može se pristupiti standardnom notacijom korišćenjem operatora tačka:

```
(*osoba).ime
(*osoba).adresa
(*osoba).starost
```

Strukturne promenljive i pokazivači

Najčešće je pogodnije funkciji poslati pokazivač na strukturu
Deklaracija

```
struct point *pp;
```

označava da je pp pokazivač na strukturu tipa struct point
Ako *pp* pokazuje na strukturu point, onda se njenim članicama može pristupiti sa (**pp*).*x* i (**pp*).*y*

```
struct point pt, *pp;

pp = &pt;
```

Ako je *p* pokazivač na neku strukturu, pristupanje članicama strukture može se kraće izvršiti korišćenjem

p->član_strukture

tako da bi poslednji red prethodnog primera glasio

```
printf("Point is (%d,%d)\n", pp->x, pp->y);
ntf("Point is (%d,%d)\n", (*pp).x, (*pp).y);
```

Strukturne promenljive i pokazivači

Ako imamo

```
struct rect r, *rp = &r;
```

onda su sledeći izrazi ekvivalentni

```
r.pt1.x
rp->pt1.x

(r.pt1).x
(rp->pt1).x
```

Nizovi struktura

Primer: Napisati program koji broji pojavljivanje pojedinih marki automobila u tekstu sa ulaza

Bez struktura

```
char *marke[MAXAUTO];
int broj[MAXAUTO];
```

Sa strukturama

```
struct automobil {
    char *marka;
    int broj;
} lista[MAXAUTO];
```

ili

```
struct automobil {
    char *marka;
    int broj;
};

struct automobil lista[MAXAUTO];
```

Nizovi struktura

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

#define MAXREC 100

struct automobil {
   char *marka;
   int broj;
} ;

int getword(char *, int);
int binsearch(char *, struct automobil *, int);
```

Nizovi struktura

```
/* binsearch: trazi marku u nizu tab[0]...tab[n-1] */
int binsearch(char *marka, struct automobil tab[], int n)
{
  int cond;
  int low, high, mid;

  low = 0;
  high = n - 1;
  while (low <= high)
  {
     mid = (low+high) / 2;

     if ((cond = strcmp(marka, tab[mid].marka)) < 0)
        high = mid - 1;
     else if (cond > 0)
        low = mid + 1;
     else
        return mid;
  }
  return -1;
}
```

Nizovi struktura

```
/* getword: get next word or character from input */
int getword(char *word, int lim)
{
  int c, getch(void);
  void ungetch(int);
  char *w = word;
  while (isspace(c = getchar()));
  if (c != EOF)
     *w++ = c;
  if (!isalpha(c)) {
     *w = '\0';
     return c;
  }
  for ( ; --lim > 0; w++)
     if (!isalnum(*w = getchar())) {
        ungetch(*w);
        break;
     }
  *w = '\0';
  return word[0];
}
```

Nizovi struktura

```
/* brojanje marki automobila */
main()
{
  int n;
  char rec[MAXREC];
    struct automobil lista[] ={
              {"Alfa Romeo", 0},
              {"Audi", 0},
              {"BMW", 0},
              {"Chevrolet", 0},
              {"Fiat", 0},
              {"Ford", 0},
              {"Honda", 0},
                      /* ... */
              {"Renault", 0},
              {"Suzuki", 0},
              {"Toyota", 0},
              {"Volkswagen", 0}
        };
```

Nizovi struktura

```
int BRAUTO = (sizeof lista / sizeof(struct automobil));
  for (n = 0; n < BRAUTO; n++)

        printf("%4d %s\n", lista[n].broj, lista[n].marka);

while (getword(rec, MAXREC) != EOF)
     if (isalpha(rec[0]))
        if ((n = binsearch(rec, lista, BRAUTO)) >= 0)
          lista[n].broj++;

  for (n = 0; n < BRAUTO; n++)
     if (lista[n].broj > 0)
        printf("%4d %s\n", lista[n].broj, lista[n].marka);

}
```

Unije

Šablon unije se opisuje na sledeći način:

```
union oznaka_unije
{
tip ime_element1;
tip ime_element2;
…
}
```

```
union primer
{
char c;
int i;
double d;
}
```

Mogu se definisati promenljive unija tipa na sledeći:

```
union primer x;
union primer a[10];
union primer *pok;
```

Unije

Pristup elementima unije se realizuje na sledeći način:

```
x.c='A';
x.i=77;
x.d=23.3;
```

U svakom momentu unija čuva vrednost samo jednog elementa, tako da program mora voditi računa o tome koji je element poslednji dobio vrednost inače se dobija besmislen rezultat.

Na primer, ako se napiše

```
x.c='Z';
double broj=35.4*x.d;
```

greška je u tome što iako je registrovana vrednost char tipa, sledeća linija računa sa elementom tipa double.

Pokazivači na unije se koriste na isti način kao pokazivači na strukture:

```
pok=&x;
double broj=pok->d;
```

Šta je rezultat rada sledećeg programa?

```
#include <stdio.h>
union primer {
    int broj;
    char slovo;
    float broj_r;
};
main(){
    union primer p;
    printf("sizeof(int)=%ld \t",sizeof(int));
    printf("sizeof(char)=%ld \t",sizeof(char));
    printf("sizeof(float)=%ld \n",sizeof(float));
    printf("sizeof(union primer)=%ld\n",sizeof(union primer));
    p.broj=5;
    printf("p.broj=%d \n",p.broj);
    p.slovo='D';
    printf("p.slovo=%c\t p.broj=%d\n",p.slovo,p.broj);
    p.broj_r=1.23;
    printf("p.broj_r=%f\tp.slovo=%c\t
p.broj=%d\n",p.broj_r,p.slovo,p.broj);
}
```

Primer definisanja unije

```
#include <stdio.h>
#include <string.h>
struct radnik {
    char prezime[20];
    char ime[20];
    char plata_ili_nadnica;
    union {
        float plata;
        struct {
            int sati_rada;
            float satnica;
        } nadnica;
    } zarada;
} osoba[20];
```

Primer definisanja unije

```
main(){
  strcpy(osoba[0].prezime,"Peric");
  printf("%s\t",osoba[0].prezime);
  osoba[0].zarada.plata=54358.5;
  printf("%.2f\n",osoba[0].zarada.plata);
  strcpy(osoba[1].prezime,"Lazic");
  printf("%s\t",osoba[1].prezime);
  osoba[1].zarada.nadnica.sati_rada=54;
  osoba[1].zarada.nadnica.satnica=850.2;
  printf("%.2f\n",osoba[1].zarada.nadnica.sati_rada*
osoba[1].zarada.nadnica.satnica);
}
```

Primeri:

Definisati strukturu prozor_vrata koja sadrži sledeće podatke navedene podatke. Vrata se prave od dasaka fiksne širine 30cm i dovoljne dužine, nadovezivanjem dasaka. Ostatak daske koji se dobije pri pravljenju vrata ne može da se upotrebi za druga vrata. Program sadrži funkciju Vrata kojom se određuje broj dasaka potreban za pravljenje svih vrata. Program sadrži i funkciju Prozor koja određuje najmanju površinu prozora. U glavnom programu učitati podatke i ispisati vrednosti dobijene iz fukcija Prozor i Vrata.

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

struct prozor_vrata{
  char tip[6];
  float sirina,visina;};


int Vrata(struct prozor_vrata vp[],int n){
    int i,d=0;
    float k;
    for(i=0;i<n;i++)
        if(!strcmp(vp[i].tip,"VRATA")) {
            k=vp[i].sirina/30;
            if(k==(int)k) d=d+(int)k;
            else d=d+(int)k+1;
        }
     return d;
}
```

```
float Prozor(struct prozor_vrata vp[],int n){
    int i=0;
    float min=0;
    while(i<n && strcmp(vp[i].tip,"PROZOR")) i++;
    if(i<n) {
       min=vp[i].sirina*vp[i].visina;
       for(;i<n;i++)
          if(vp[i].sirina*vp[i].visina<min)
min=vp[i].sirina*vp[i].visina;
    }
    return min;
}
```

```
main()
struct prozor_vrata vp[100];
int i,n=0;
char s[6];
float s1,v1;
 scanf(“%s”,s);
while(s!=‘0’){
      strcpy(vp[n].tip,s);
      scanf("%f%f",&s1,&v1);
      vp[n].sirina=s1;
      vp[n++].visina=v1;

      scanf(“%s”,s);
   }
}

printf("%d\n",Vrata(vp,n));
printf("%.2f\n",Prozor(vp,n));
}
```