C programavimo kalba

10 paskaita
(Struktūros)

Struktūrų sintaksė

Struktūra – tai vienodo arba skirtingo tipo kintamųjų rinkinys.

Sintaksė:

```
struct vardas
{
   type1 var1;
   type2 var2;
    ......
   typeN varN;
} ;
// Gale kabliataškis ;
    būtinas!!!!
```

```
struct CD
    {
       char name[20];
       char description[40];
       char category[12];
       float cost;
       int number;
    } ;                          //  sukuriam struktūrą

struct CD  kortele1;   // struktūros kintamasis
struct CD disc[10], cdrom [100];
struct CD *ptr;
```

Struktūros elementų inicializacija

```
struct CD
 {  char id[4];
    char name[20];
    char description[40];
    char category[12];
    float cost;
    int number;
 } disc = {“p12”, “Aliukai”,“Best Hits”,”pop-muzika”, 12.20, 12};

struct CD disc1={"p13”, “Ventukai”,“Top Hits”,”pop-folk”, 10.50, 10};

struct CD disc2;

puts (“Iveskite koda: “);    fgets(disc2.id, 4, stdin);
strcpy(disc2.name, “Ozzy Osborne”);
puts (“Iveskite albumo pavadinima”);   gets(disc2.description);
strcpy(disc2.category, “rock“);
disc2.cost = 2.5f;
disc2.number = 4;
```

Bitų laukai struktūroje

Siekiant taupyti atmintį, struktūros elementams gali būti priskirti bitų laukai. Tuomet apibrėžiant struktūrą, jos elementams atmintyje išskiriama tam tikras bitų skaičius, paprastai mažesnis nei baziniams duomenų tipams.

Sintaksė:

struct Laukas {
Lauko_deklaravimas : bitų_skaičius; };

```
struct DATE
{
    unsigned short Year;
    unsigned short Month;
    unsigned short Day;
}
```

Atmintyje 6 baitai = 48 bitai

```
struct DATE
{
unsigned Year : 12;
unsigned Month : 4;
unsigned Day : 5;
}
```

Atmintyje 21 bitas

Struktūros – funkcijų argumentai

Struktūros laikomos vartotojo sukurtu duomenų tipu, todėl kaip ir bet kuris bazinis duomenų tipas taip ir struktūros gali būti perduotos per funkcijos argumentų sąrašą. Perdavimas atliekamas kopijuojant reikšmes.

```
struct namas
 {   int nr;
     int aukstai;
     char gatve[40];
};
```

Funkcijos prototipas

```
void Func(struct namas);
```

Funkcija taip pat gali grąžinti struktūros tipo objektą. Tuo atveju struktūros duomenys perduodami išviečiančiai funkcijai. Pavyzdžiui funkcijos prototipas:

```
struct namas Func1(int, float);
```

```
struct namas Func1 (int a, float b);
```

Realizacija

Pavyzdys su funkcija

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

struct filmai_t {
                    char title [50];
                    int year;
                  } mano, tavo;

void printmovie (struct filmai_t );

void main ()
{  char buffer [50];
   strcpy (mano.title, "2001 A Space Odyssey");
   mano.year = 1968;
   printf( "Ivesk pavadinima: ");
   fgets (tavo.title, 50, stdin);
   printf( "Ivesk metus: ");
   scanf("%d", &tavo.year);
   printf("Mano filmas:\n ");
     printmovie (mano);
   printf("Tavo filmas:\n ");
     printmovie (tavo);
   return 0;
}

void printmovie (filmai_t movie)
{
  puts(movie.title);
  printf( " ( %d )\n", movie.year);
}
```

Pavyzdys su funkcija

```
#include <stdio.h>
struct CD
     {
     char name[20];
     float price;
     int number;
     } disc, getdisc( );

void main()
   {  disc = getdisc();
   printf ( "Ivesta informacija apie diska \n");
   printf ( "Pavadinimas, %s\n", disc.name);
   printf ( "Kaina %d\n", disc.price);
   printf ( "Kiekis %d\n", disc.number);
   }
```

```
struct CD getdisc( )
   {
   struct CD in_disc;
   puts("Disko pavadinimas: ");
   fgets(in_disc.name, 20, stdin);
   puts("Disko kaina: ");
   scanf("%d", &in_disc.price);
   puts("Disku kiekis: ");
   scanf("%d", &in_disc.number);
    return (in_disc);
   }
```

Struktūrų masyvas

Dažniausiai naudojamos ne pavienės struktūros, o jų masyvai, kurie sudaro duomenų bazių pagrindą. Struktūrų masyvai apibrėžiami analogiškai, kaip ir įprastiniai bazinių duomenų tipų masyvai.

```
struct CD
{    char name[20];
     float price;
     int number;
}
```

Struktūrų masyvas

struct CD disc[10];

```
     strcpy(disc[0].name, "Aliukai");
     strcpy(disc[1].name, "Broliukai");
     ..............
     printf("Mano CD kolekcija:\n");
      for (int i = 0; i < 10; i++)
             { printf( "Pavadinimas: %s\n", disc[i].name);
       ........
      }
```

Rodyklės ir struktūros

Kaip ir bet kuriam duomenų tipui, taip ir struktūrai gali būti sukurta rodyklė. Taip pat struktūra gali būti rodyklės tipu.

Rodyklės deklaravimas

struct Struktūros_pavadinimas * rodyklės_pavadinimas;

Pavyzdžiui:

```
struct movies_t
{   char title [50];
    int year; };
 struct movies_t  filmas, *ptr_filmas;

 ptr_filmas = &filmas;      // adreso priskyrimas
```

Struktūros elementai, kai naudojama rodyklė, pasiekiami naudojant ne ".", o "**->**".

Pavyzdžiui: ptr_filmas->title; ptr_filmas->year;

Pavyzdys su rodyklėmis

```
#include <stdio.h>
 struct movies_t  { char title [50]; int year;  };

void main ()
{   char buffer[50];
     struct movies_t  amovie;
     struct movies_t  *pmovie;                  // sukuriame rodyklę
     pmovie = & amovie;                         // priskiriame adresą

     puts ( "Ivesk filmo pavadinima: ");
          gets (pmovie->title);                 // priskiriame reikšmę
     puts ( "Ivesk filmo metus: ");
         gets (buffer);
    pmovie->year = atoi (buffer);               // keičiame į int

    puts ("\nTu ivedei:\n");
       printf( "%s ", pmovie->title);           // tas pats (*pmovie).title
    printf (" ( %d )\n", pmovie->year );
}
```

Paaiškinimai

Išraiška	Paaiškinimas	Ekvivalentas
pmovie.title	Elementas **title** iš struktūros **pmovie**	
pmovie->title	Elementas **title** iš struktūros į kurią rodo rodyklė **pmovie**	(*pmovie).title
*pmovie.title	Rodyklės **title** iš struktūros **pmovie** reikšmė	*(pmovie.title)

Rodyklės į struktūras gali būti naudojamos funkcijų argumentų sąraše. Tai žymiai pagreitina duomenų perdavimą, lyginant su kopijavimu. Sintaksės prasme struktūros rodyklių naudojimas, kaip funkcijos argumento, niekuo nesiskiria nuo kitų rodyklių naudojimo. Pavyzdžiui:

```
struct HOUSE
{ int A; float B;} *ptr;

void Func1(struct HOUSE *ptr);
```

Struktūra struktūroje

Struktūra gali būti kitos struktūros elementu. Pavyzdžiui:

```
struct movies_t
{ char title [50];
   int year; };

struct friends_t
{ char name [50];
   char email [50];
   struct movies_t  favourite_movie; } charlie, maria;

struct friends_t * pfriends = &charlie;
```

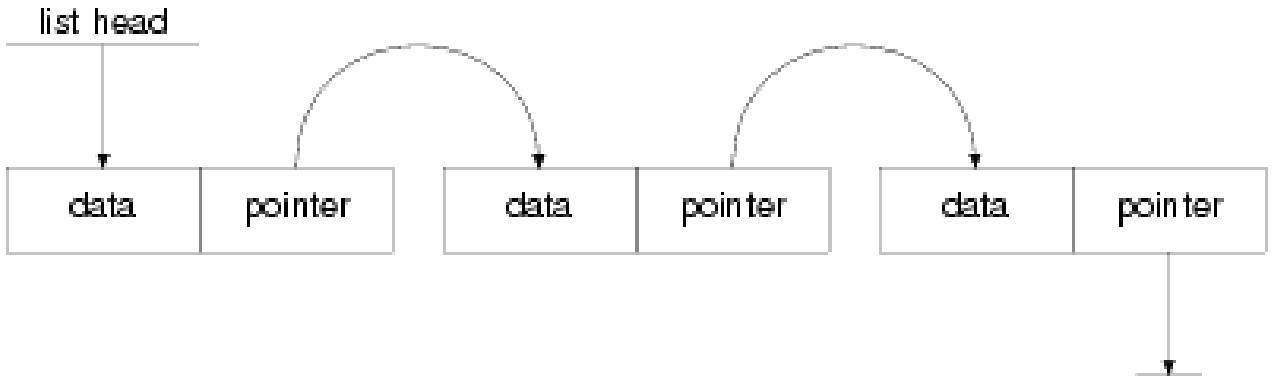
Su šiomis struktūromis galima naudoti tokias išraiškas:

```
charlie.name
maria.favourite_movie.title
charlie.favourite_movie.year
pfriends->favourite_movie.year
```

} *tai tas pats elementas*

Sąrašai

Struktūros ir rodyklės naudojamos surištuose sąrašuose (linked lists).



Toks sąrašas turi būti deklaruojamas taip:

```
struct list_ele
{   int data;                    // duomenų laukas
    struct list_ele *ele_p;   // rodyklė, skirta surišti struktūrą iš dešinės
};
```

Naudojamas nepabaigtas struktūros deklaravimo tipas.

Sąrašo pavyzdys

```
#include <stdio.h>
#include <stdlib.h>
struct list_ele{
      int data;
      struct list_ele *pointer;
} ar[3];
void main(){
      struct list_ele *lp;
      ar[0].data = 5;
      ar[0].pointer = &ar[1];
      ar[1].data = 99;
      ar[1].pointer = &ar[2];
      ar[2].data = -7;
      ar[2].pointer = NULL;                  /* NULL žymi sąrašo pabaigą */

      lp = ar;
      while(lp){
            printf("contents %d\n", lp->data);
            lp = lp->pointer;  }            // keičiama rodyklės reikšmė
       }
```

Junginiai (union)

Junginiai (union) - tai tam tikros struktūros, kurios naudojamos tuomet, kai norima taupyti atmintį, nes toje pačioje atminties srityje talpinami skirtingų junginių laukai. Tai reiškia, kad tam tikru laiko momentu atmintyje gali būti **tik vienas** iš apibrėžtų junginio laukų.

```
#include <stdio.h>
#include <stdlib.h>
main(){
      union testas {
            float var_f;
            int var_i;
      } uno;
      uno.var_f = 23.5;
      printf("Reiksme = %f\n", uno.var_f);   // išvesti var_i neįmanoma
      uno.var_i = 5;
      printf("Reiksme %d\n", uno.var_i);     // išvesti var_f neįmanoma
 }
```