C programavimo kalba

5 paskaita

(Funkcijos, masyvai)

Funkcijų pavyzdys

```
// Skaičių lyginimo programa

#include <stdio.h>
 void pmax(int, int);    /* prototipas */

int main()
   {int i, j;
        for (i = -10; i <= 10; i++)
            { for (j = -10; j <= 10; j++)
                  pmax( i, j);
             }
     return 0;
   }
```

```
// Funkcijos tekstas

    void pmax(int a1, int a2)
    {  int biggest;
        if (a1 > a2)
            biggest = a1;
        else
            biggest = a2;
      printf("Is skaiciu %d ir %d didesnis
                  %d\n", a1, a2, biggest);
     }
```

Kintamųjų modifikatoriai

Modifikatorius	Taikymas	Veikimo sritis	Gyvavimo laikas
Auto	Lokalus	Blokas	Laikinas
Register	Lokalus	Blokas	Laikinas
Extern	Globalus	Blokas	Laikinas
Static	Lokalus/Globalus	Failas	Pastovus
volatile	Globalus	Failas	Pastovus

Modifikatorius auto

Modifikatorius ***auto*** naudojamas aprašant lokalius kintamuosius, tačiau jis dažnai praleidžiamas, nes pagal nutylėjimą lokalūs kintamieji ir yra auto tipo.

Pavyzdys

```
#include <stdio.h>
int main ()
{
      auto int MyVar = 2;          // tas pats kas kaip int MyVar = 2;
      printf ("Kintamasis MyVar = %d\n", MyVar );
      return 0;
}
```

Modifikatorius register

Šis modifikatorius talpina lokalų kintamąjį į procesoriaus registrus, tačiau tai ne visad pavyksta, kadangi procesoriaus registrų skaičius ribotas. Nesekmės atveju kintamasis tampa *auto* tipo kintamuoju.

Paprastai modifikatorius **register** naudojamas su *int* ir *char* tipo kintamaisiais. Jei naudojami kiti kintamųjų tipai, jie saugomi procesoriaus spartinančiojoje atmintyje (cache).

Pavyzdys

```
#include <stdio.h>
int main ()
{
      register int MyVar = 2;
      printf ("Kintamasis MyVar = %d\n", MyVar );
      return 0;
}
```

Modifikatorius extern

Jei programa sudaro keletas modulių (funkcijų esančių skirtinguose failuose), kai kurie kintamieji gali būti naudojami duomenų perdavimui iš vieno modulio kitam.

Tam reikia:

- viename modulyje (faile) apibrėžti kintamąjį kaip globalų;
- kitame modulyje (faile) apibrėžti tą patį kintamąjį kaip ***extern***;

```
// Failas my.h

   extern bool zyme;

   void keisti (void)

     {
         zyme != zyme;
      }
```

```
// Failas program.c
   #include "my.h"
   #include <stdio.h>
     bool zyme = true;

   void main (void)

     { keisti ();
       if (zyme) printf ("False\n");
      }
```

Extern modifikatoriaus pavyzdys

```
/* Pirmasis failas */

int i;
void f_in_other_place (void);

int main ()
{ i = 0;
  f_in_other_place ();
   return 0; }

/* pirmo failo pabaiga */
```

```
/* Antrasis failas */
extern int i;

void f_in_other_place (void)
{ i++; }

/* Antro failo pabaiga */
```

Statiniai kintamieji

Statiniai kintamieji nenaikinami funkcijai baigus darbą. Jei apibrėžiami žodžiu static. Jei apibrėžimo metu, toks kintamasis nebuvo inicializuotas, jam priskiriama reikšmė 0.

Jei statinis kintamasis apibrėžiamas globaliai, jis egzistuoja visos programos vykdymo metu, o jo matomumas užtikrinamas visoje programoje.

Jei statinis kintamasis apibrėžiamas lokaliai, jis taip pat egzistuoja visos programos vykdymo metu, tačiau jo matomumas užtikrinamas tik tos funkcijos, kurioje jis yra apibrėžtas ribose.

Statiniai kintamieji negali būti apibrėžti kituose failuose, kaip išoriniai.

Static modifikatoriaus pavyzdys

```
#include <stdio.h>
int count (void);
int main ()
{  int rezult;
   for (int i = 0; i< 30; i++)
   {   rezult = count ();
        printf (" Funkcija buvo iskviesta %d karta \n", rezult);
    }
  return 0;
}
```

```
int count (void)
{ static short counter = 0;
   counter ++;
   return counter;
}
```

Inline funkcijos

Jei programoje dažnai iškviečiama ta pati funkcija ir ji nėra didelės apimties, tikslinga tokia funkcija apibrėžti kaip **inline**. Tai reiškia, kad tokios funkcijos tekstas bus įtrauktas į programos tekstą, todėl bendras programos vykdymo laikas sutrumpės, nes nebus prarandamas laikas iškvietimo procedūroms atlikti.

Pavyzdys:

```
#include <stdio.h>
inline int sum (int, int);
```

```
int sum (int x, int y)
{  return x+y;  }
```

```
int main()
{  int A = 2, B = 4, C =5;
      printf ( "Skaiciu %d ir %d suma %d\n", A, B, sum (A, B) );
      printf ( "Skaiciu %d ir %d suma %d\n", C, B, sum (C, B) );
      printf ( "Skaiciu %d ir %d suma %d\n", A, C, sum (A, C) );
   return 0;
}
```

Matematinės funkcijos

Matematinės funkcijos saugomos bibliotekos faile math.h

Paskirtis	Prototipas	Paaiškinimai
Laipsninė funkcija	double pow (double x, double y)	x^y
Logaritminė funkcija (natūrinis) dešimtainis	double log (double x) float logf(float x) long double logl (long double x) double log10 (double x) float log10f(float x) long double log10l (long double x)	ln x $\log_{10}x$
Kvadratinės šaknies traukimas	double sqrt (double x) float sqrtf(float x) long double sqrtl(long double x)	
Modulis	int abs (int x) float fabsf (float x) double fabs (double x) long labs(long x) long double fabsl(long double x)	|x|

Matematinės funkcijos (tęsinys)

Paskirtis	Prototipas	Paaiškinimai
Trigonometrinės funkcijos	double cos (double x) float cosf(float x) double sin (double x) float sinf(float x) double tan (double x) float tanf(float x) double acos (double x) float acosf(float x) double asin (double x) float asinf(float x) double atan (double x) float atanf(float x)	cos(x) sin(x) tg(x) arccos(x) arcsin(x) arctg(x)
Apvalinimo funkcijos	double floor(double x) long double floor(long double x) double ceil(double x) long double ceil (long double x)	Apvalinimas į mažesnę pusę Apvalinimas į didesnę pusę
double fmax(double x, double y); double fmin(double x, double y);double exp(double x); **double modf (double** *x* **, double *** *ipart* **) –** suskaido x į sveiką(*ipart*) ir trupmeninę dalį (grąžinama reikšmė)		

Apvalinimo pavyzdys

```
#include <stdio.h>
#include <math.h>
double round (double);
void main()
{ double x = 20.45,  y = 5.3;
   double sandauga;
   sandauga = x * y;
   printf( "Suapvalinta sandauga %g\n", round(sandauga) );
}
```

```
double round (double num)
{ double fract, value;
   fract = modf (num, &value);   // prototipas  double modf(double x, double *i)
   if (fract < 0.5)
       num = value;
   else
       num  = value+1;
   return num; }
```

Masyvai

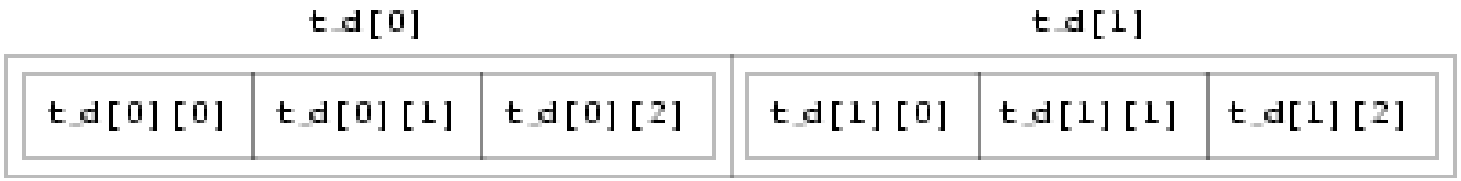
Vieno tipo duomenų rinkinys vadinamas masyvu.

Masyvo elementai gali būti int, float, double, char, short, long, long double t. y. bazinių tipų duomenys taip pat ir išvestinių tipų duomenys.

Masyvai būna vienmačiai, dvimačiai trimačiai ir t.t.

Atmintyje masyvo elementai saugomi nuosekliai vienas po kito.

Dvimačiai masyvai saugomi eilutėmis:



Masyvai

Masyvų elementai numeruojami. Numeracija pradedama nuo **0**.

int array[10]; // *dešimties sveikų skaičių masyvas*

float masyvas[14]; // *keturiolikos trupmeninių skaičių masyvas*

Pradinių reikšmių priskyrimas masyvo elementams (inizializacija)

```
int arr[3] = { 12, 4, 5 };    // pradinių reikšmių priskyrimas masyvo elementams
float mas[3] = { 1.2, 3.5, 5.5};
int array [5] = {1, 3, 5, }    // dalinė inicializacija; array[3]=0; array[4]=0

y = arr[0];       // y priskiriama masyvo pirmo elemento reikšmė
z = arr[1];       // z priskiriama masyvo antro elemento reikšmė
```