C programavimo kalba

3 paskaita

(Sąlygos ir ciklo operatoriai, funkcija scanf())

Sąlygos operatorius *if - else*

Sąlygos operatoriai skirti perduoti programos vykdymą vienai ar kitai programos šakai.

Operatorius ***if*** paskirsto programos vykdymą priklausomai nuo sąlygos.

Sintaksė:

```
if (sąlyga)
    operatorius1
else
    operatorius2
```

Jei sąlyga turi reikšmę *true* arba nenulinę reikšmę, vykdomas *operatorius1*, jei salyga turi reikšmę *false* arba lygi 0, vykdomas *operatorius2*.

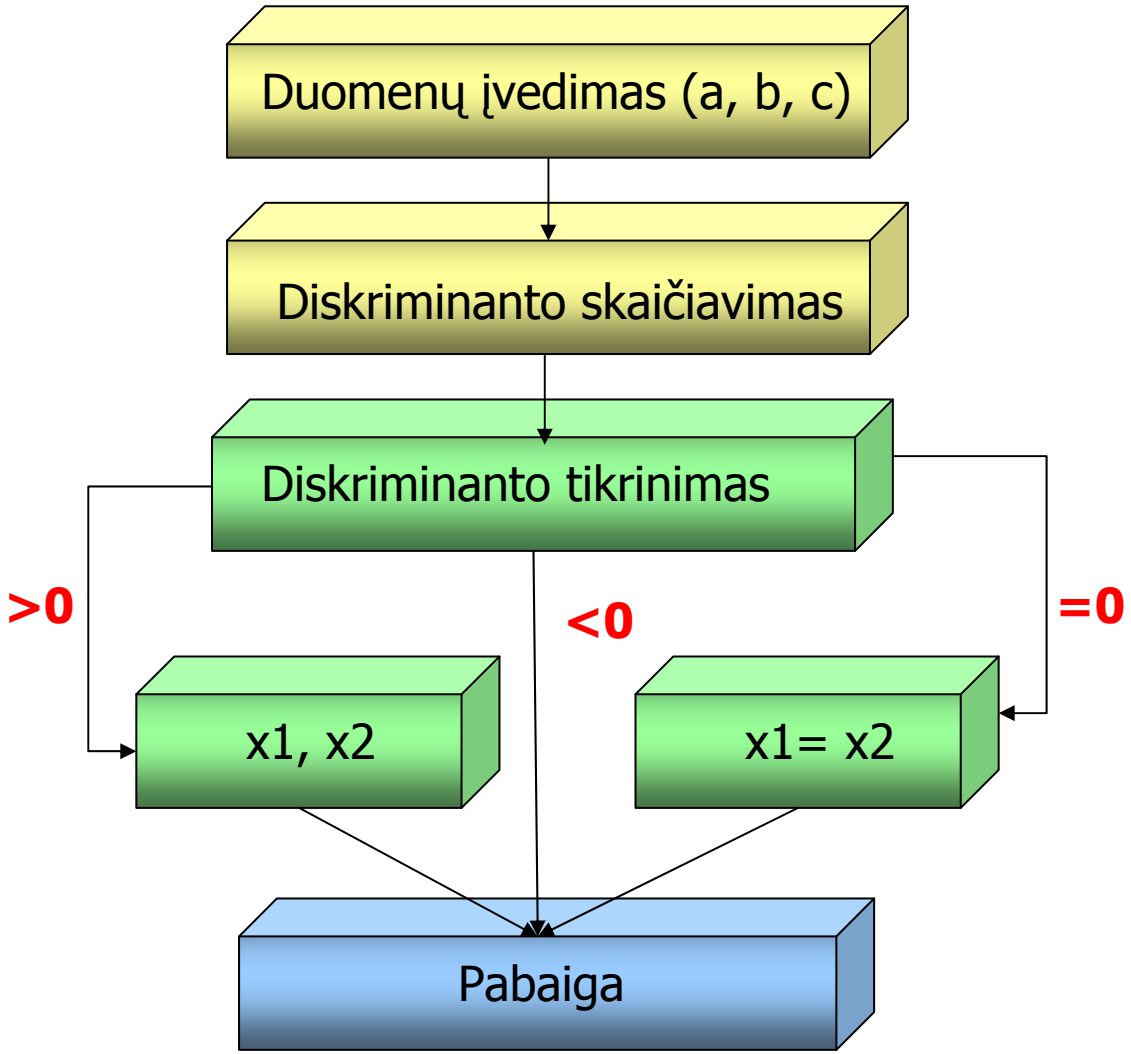
Trumpa forma:

```
if (sąlyga)                vietoj        if ( sąlyga != 0 )
    .....                                    .....
```

Pavyzdys

```
if (a > b)
    z = a;
else
    z = b;
```

```
if (n > 0)
{
   if (a > b)
        z = a;
}
else
    z = b;
```



Pavyzdys

```
#include <stdio.h>
#include <math.h>
int main() {
   int a = 1, b=2, c = -3;
   float D, x1, x2;
D = (float) b*b – 4*a*c ;
if ( D < 0 )
   {printf (" Lygtis neturi racionaliu saknu...\n");
     return 0; }
if ( D == 0 )
  { x1 = x2 = (float) -b/(2*a);
     printf("Saknis x= %5.3f \n",  x1);
   }
else
{  x1 = (float) (-b+sqrt(D) )/ (2*a);
    x2 = (float) (-b-sqrt(D) )/ (2*a);
  printf("Saknis x1= %5.3f, saknis x2 =  %5.3f  \n",
           x1, x2); }
  return 0; }
```

Sąlygos operatorius *?:*

Jei operatoriai ir sąlyga yra pakankamai paprasti, galima naudoti sąlygos operatorių **?:**

Sintaksė:

sąlyga ? *operatorius1* : *operatorius2*;

Jei sąlyga teisinga (*true*), vykdomas *operatorius1*, jei ne (*false*) – *operatorius2*;

Pavyzdys:

```
int a=10, b=20;
int max;
    max = (a > b) ? a : b;          //  (a > b) ?  max = a: max = b;
      printf( "Max = %d \n", max);
```

else if konstrukcija

else if konstrukcija naudoja esant keletai sąlygų t.y. keletui variantų.

```
if (sąlyga1)
   išraiška1
else if (sąlyga2)
  išraiška2
else if (sąlyga3)
  išraiška3
else
  išraiška4
```

```
int a = 1, b=2, c = -3;
float D, x1,x2;
D =(float) b*b – 4*a*c;
if ( D < 0 )
   {printf (" Lygtis neturi racionaliu saknu...\n");
     return 0; }
else if ( D == 0 )
  { x1 = x2 = (float) -b/(2*a);
   printf("Saknis x= %f \n", x1);
   }
else
{ x1 = (float) (-b+sqrt(D)) / (2*a);
   x2 = (float) (-b-sqrt(D)) / (2*a);
  printf("Saknis x1= %f, saknis x2 = %f  \n",
           x1,x2);  }
 }
```

Operatorius *switch*

Jei sąlyga gali priimti keletą reikšmių, naudojamas daugiavariantinis operatorius switch.

Sintaksė:

```
switch (sąlyga)
   { case konstantinė_išraiška1: operatorių grupė;
     break;
     case konstantinė_išraiška2: operatorių grupė;
     break;
     default: operatrių grupė
   }
```

Pavyzdys:

```
char answer = 'y';
switch (answer)
{ case 'y':  printf ("Darbas tesiamas... \n");
   break;
   default: printf( "Viso gero... \n");
}
```

Ciklo operatorius *for*

Ciklo operatorių pagalba galima tą pačią programos kodo dalį (iteraciją) daug kartų.

Sintaksė:

```
for (expr1; expr2; expr3)
    operatorius_arba_ ju_blokas;
```

expr1; *expr2*; *expr3* – ciklo valdymo išraiškos.
expr1 – paprastai ciklo valdymo kintamojo inicializacija.
expr2 – paprastai sąlyga, tikrinanti ciklo kintamojo reikšmę.
expr3 - paprastai ciklo kintamojo prieaugio operatorius

Begalinis ciklas

```
for ( ; ; )
    printf( "Begalinis ciklas... \n");
```

for operatoriaus pavyzdys

Skaičių sumos nuo 10 iki 19 (20 neįeina) skaičiavimas

```
#include <stdio.h>
int main ()
{ int Sum = 0;
    for (int i = 10; i < 20; i++ )
       Sum += i;
    printf ("Suma skaiciu nuo 10 iki 20 bus %d \n", Sum);
  return 0;
}
```

```
// Galimas variantas

int Sum1= Sum2 = 0;
for (int i = 10, j =50 ; i < 20; i++, j++ )
       { Sum1 += i;
         Sum2 += j;
        }
```

Pirminio skaičiaus paieška

```
#include <stdio.h>

void main()
{   int i,j;
    bool yes;

      for ( i=1; i< 100; i++ )
      {   yes = true;
          for ( j=i;  j>2;  j-- )
          {    if ( i % (j - 1) == 0)
                    yes = false;
          }
          if (yes)
              printf("%d \n", i);
      }
}
```

Pavyzdys

```
#include <stdio.h>
void main() {
    int i;
    for(i=3; i<13; i++) {
    switch (i) {
                case 3 : printf("Trys\n");
                    break;
                case 4 : printf("Keturi\n");
                    break;
                case 5 :
                case 6 :
                case 7 :
                case 8 : printf("Penki - astuoni\n");
                    break;
                case 11 : printf("Vienuolika\n");
                    break;
                default : printf("Neaprasyta reiksme\n");
                    break;
} } }
```

Ciklo operatorius *while*

Sintaksė:

```
while (sąlyga)
    operatorius arba jų blokas;
```

Jei sąlyga turi reikšmę *true* arba tai *nenulinis skaičius*, vykdomas ciklas.
Jei sąlyga konstanta – ciklas begalinis.

Pavyzdys:

```
main()
{ int counter = 10, Sum = 0;
   while ( counter)
    {  Sum +=counter;
       counter - - ;
     }
 printf( "Suma nuo 0 iki 10 lygi %d \n",  Sum);
 return 0;
}
```

Ekrane

Suma nuo 0 iki 10 lygi 55

Ciklo operatorius *do-while*

Skirtingai nei ciklo operatorius *while*, šis ciklo operatorius pirmiausiai vykdo operatorių bloką, o tik po to atliekamas sąlygos tikrinimas. Tokia konstrukcija garantuoja, kad ciklo turės bent vieną iteraciją.

Sintaksė:

```
do
   operatorius_arba_ju_blokas;
while (sąlyga)
```

Pavyzdžiai:

```
char answer;
do
{ printf (“Tesime darba? (y/n):”);
  scanf (“%c”, &answer);
 }
while (answer != ‘N’ || answer != ‘n’)
```

```
#include <stdio.h>
  void main()
  {  int i = 0;
      do {
           printf("i = %d\n", i);
           i++;
       } while(i < 5);
}
```

Funkcija *scanf()*

Reikalingas antraštės failas: stdio.h

Tai funkcija, skirta formatuotam duomenų nuskaitymui iš klaviatūros. Nuskaityti duomenys saugomi kintamuosiuose, kurie nurodyti argumentų sąraše.

Sintaksė: int scanf (const char * *format* [, *argument* , ...]);

kur *format* – skaityti sekančią skaidrę.

argument – tai nuorodos į kintamuosius, kuriems priskiriamos atitinkamos reikšmės įvestos iš klaviatūros pagal nurodytą formatą.

Kintamųjų formatų skaičius ir argumentų (nuorodų į kintamuosius) skaičius turi sutapti.

Funkcija grąžina sėkmingai nuskaitytų duomenų skaičių. Priešingu atveju – neigiamą skaičių.

format prototipas:

%[*][*width*][*modifiers*]*type*

format aprašymas

Eilutė, kuri aprašo duomenų nuskaitymo formatą ir yra vienas iš žemiau nurodytų punktų:

Tarpo simboliai: funkcija nuskaito ir ignoruoja tarpo, naujos eilutės arba tabuliacijos simbolį, kuris sutinkamas prieš sekantį netarpo simbolį. Tarpo simbolių gali būti bet koks skaičius.

Simboliai (ne tarpo, naujos eilutės, tabuliacijos arba prasidedantys % simboliu): funkcija nuskaito ir atmeta bet kurį simbolį, kuris sutampa su duotuoju. Jei toks simbolis nerastas – grąžinama klaida.

Formato simboliai: tai keletas simbolių, kurie pradedami %. Jie nurodo formatą, kuriuo turi būti nuskaityti duomenys ir priskirti atitinkamam kintamajam, nurodytam *argument* dalyje.

Pavyzdys

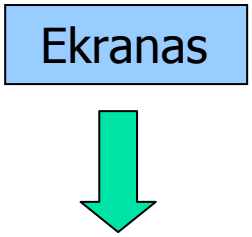
```
/* scanf () pavyzdys */
#include <stdio.h>
int main ()
 { char str [80];
    int i;
    printf (“Ivesk savo varda: ");
    scanf ("%s", str);

    printf (" Ivesk savo amziu : ");
    scanf ("%d", &i);

    printf ("Mr. %s , %d metu.\n", str, i);

    printf (" Ivesk savo sesioliktaini skaiciu: ");
    scanf ("%x", &i);

    printf (“Tu ivedei %#x (%d).\n",i,i);
    return 0; }
```



Ivesk savo varda : **Albinas**

Enter your age: **23**

Mr. Albinas, 23 metu.

Ivesk savo sesioliktaini skaiciu: **ff**

Tu ivedei 0xff (255).

format dalys

*****	Duomenys nuskaitomi, bet ignoruojami. Jie nepriskiriami *argumentui* (t.y kintamajam).
width	Nurodo maksimalų nuskaitomų simbolių skaičių.
modifiers	Apibrėžia skirtingą duomenų dydį, priskiriamą *argumentui*: •**h** : **short int** •**l** : **long int** (jei integer) arba **double** (jei floating point). •**L** : **long double**
type	Simboliai, nurodantys duomenų tipą bei kaip jie bus nuskaityti.

type reikšmės

type	**Reikšmė**	Reikalaujamas duomenų tipas
c	**Vienas simbolis**: nuskaito vieną simbolį tame tarpe ir tarpo simbolį.	char *
d	**Dešimtainis skaičius**: skaičius tame tarpe ir – simbolis.	int *
e,E,f,g,G	**Trupmeninis dešimtainis skaičius**: paprastoje arba eksponentinėje formoje. Galimi variantai -732.103 arba 7.12e4	float *
o	**Aštuntainis skaičius**.	int *
s	**Simbolių eilutė.** Nuskaitoma simbolių seka iki sutinkamas tarpas, naujos eilutes arba tabuliacijos simbolis.	char * (string)
u	**Beženklis dešimtainis skaičius**	unsigned int *
x	**Šešioliktainis skaičius**	int *

scanf() pavyzdys

```
#include <stdio.h>
int main ()
{int a, b;
  float x, y;
  printf ("Iveskite du sveiko tipo skaicius:> ");
        scanf("%d %d", &a, &b);
  printf ("Siu skaiciu sandauga: %d \n",  a*b);
  printf ("Iveskite du trupmeninius skaicius:> ");
        scanf ("%f  %f", &x, &y);
  printf ("Siu skaiciu suma: %6.3f ",  x+y);
  printf (", o skirtumas: %e \n", x-y);
  return 0;
}
```

Į ekraną

Iveskite du sveiko tipo skaicius:> **3 5**
Siu skaiciu sandauga: 15
Iveskite du trupmeninius skaicius: **1.3 5.2**
Siu skaiciu suma: 6.5 , o skirtumas -3.9e+000