# C programavimo kalba

**2 paskaita**

**(Operatoriai, funkcija printf() )**

# Konstantos (tęsinys)

**Specialios valdymo simbolinės konstantos**
Egzistuoja simbolinės konstantos, kurių neimanoma tiesiogiai įvesti klaviatūra, pvz. tabuliacija, naujos eilutės ir t.t

| Kodas | Reikšmė |
|---|---|
| \b | Kursoriaus grąžimas per vieną poziciją atgal |
| \f | Perėjimas į naują puslapį |
| \n | Perėjimas į naują eilutę |
| \r | Kursoriaus grąžinimas į einamos eilutės pradžią |
| \t | Horizontali tabuliacija |
| \“ | Dvigubos kabutės |
| \‘ | Apostrofos simbolis |
| \\ | Pasvirusio brūkšnio simbolis |
| \v | Vertikali tabuliacija (į apačią) |

# *enum* operatorius

Esant dideliam konstantų skaičiui, patogu priskirti joms reikšmes naudojant operatorių **enum** (iš angl. *enumerate*).

**Sintaksė**: enum Name { item1 [=reikšmė1], item2 [=reikšmė2]…}

**Pavyzdys:**

```
const int RED = 0;
const int YELLOW = 1;
const int GREEN = 2;
// Panaudodami enum tai užrašome trumpiau
enum COLOR { RED, YELLOW, GREEN};
/* pagal nutylėjimą pirmasis  =0, kiti didinami vienetu */

// Galimas ir toks variantas
 enum COLOR { RED=19, YELLOW=12, GREEN}; // GREEN = 13
```

# Duomenų tipų keitimas

Veiksmus galima atlikti tik su vienodo tipo duomenimis, todėl skirtingo tipo duomenis reikia suvienodinti.

Galimas du duomenų tipų keitimo variantai:

- išreikštinis
- neišreikštinis

**Pavyzdys**

```
int Integer =54;
float floating = 13.623;
Integer = (int) floating + 12;   // išreikštinis tipo keitimas Ats: 25
floating = floating + 11;
Integer = floating;              // neišreikštinis tipo keitimas Ats: 13
floating = floating + 12.4f;
```

# Aritmetinės operacijos

Manipuliacijos duomenimis atliekamos taikant aritmetines operacijas. Šios operacijos gali būti atliekamos su vienu (unarinė) arba dviem (binarinė) operandais.

| Operacija | Reikšmė |
|---|---|
| + | Sudėtis |
| - | Atimtis |
| * | Daugyba |
| / | Dalyba |
| % | Liekana dalybos metu (9%2=1) |
| = | Priskyrimo operatorius |
| += | Priskyrimas ir sudėtis |
| -= | Priskyrimas ir atimtis |
| *= | Priskyrimas ir daugyba |
| /= | Priskyrimas ir dalyba |
| %= | Piskyrimas ir liekana |
| ++, -- | Didinimas 1, mažinimas 1 |

# Pavyzdys

```
int a = 0, b = 4, c = 90;
char z = '\n';
a = b;
a = b + c;
a = b -  2;
a = b*3;
a =c/(b+6);
a= 5%2;
a +=b;          //a = a + b
a*=c-50;        //a = a * (c-50)
a-=60;          //a = a - 60
a%=8;           //a = a % 8
```

# Lyginimo operacijos

Lyginimo operatoriai skirti palyginti dviejų kintamųjų reikšmes. Lyginimo operatorių atsakymas true arba false.

| Operatorius | Reikšmė |
|---|---|
| < | Mažiau |
| <= | Mažiau arba lygu |
| > | Daugiau |
| >= | Daugiau arb lygu |
| = = | Lygu |
| != | Nelygu |

# Loginės operacijos

| Operacija | Reikšmė |
|---|---|
| & & | loginis IR (daugyba) |
| \| \| | Loginis ARBA (sudėtis) |
| ! | Loginis NE (inversija) |

| A | B | C=A& & B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

| A | B | C=A\| \| B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

| A | C !=A |
|---|---|
| 0 | 1 |
| 1 | 0 |

# Operacijų prioritetai

C kalboje operatoriai vykdomi tam tikra tvarka. Pavyzdžiui matematiniai veiksmai atliekami iš kairės į dešinę, o priskyrimo operatoriai iš dešinės į kairę.

Operatoriai taip pat turi prioritetus, kurie apsprendžia jų vykdymo eiliškumą. Kaip ir matematikoje, operatorius galima grupuoti (). Sugrupuoti operatoriai turi aukščiausią prioritetą.

**Operacijų prioritetai:**

1. Loginiai
2. Priskyrimo
3. Lyginimo
4. Aritmetiniai
5. Inkremento, dekremento

```
// Pavyzdys
int x= 0, y=0;
int a=3, b=34, c = 82;
x= a*b*c;
y = (a*(b+c));
y = a++;
b = c--;
```

# Funkcija *printf()*

**Reikalingas antraštės failas:** stdio.h

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

kur *format* – tai eilutė, kurioje yra išvedamas į ekraną tekstas bei formatavimo simboliai (pradedami % ženklu).

*argument* – tai kintamųjų vardai, kurių reikšmės yra atitinkamai formatuojamos išvedant į ekraną.

Formatavimo simbolių ir argumentų skaičius turi sutapti.

Funkcija grąžina išvestų į ekraną simbolių skaičių, jei ji sėkmingai įvykdyta. Priešingu atveju – neigiamą skaičių.

---

*format* prototipas:

**%**[*flags*] [*width*] [**.***precision*] [*modifiers*] *type*

# *type* reikšmės

| *type* | Reikšmė | Pavyzdys |
|---|---|---|
| **c** | Simbolis | **a** |
| **d** arba **i** | Dešimtainis sveikas skaičius | **392** |
| **e** | Eksponentinė forma naudojant **e** simbolį | **3.9265e2** |
| **E** | Eksponentinė forma naudojant **E** simbolį | **3.9265E2** |
| **f** | Dešimtainis trupmeninis skaičius | **392.65** |
| **g** | Naudojamas trumpesnis iš dviejų formatų: %e arba %f | **392.65** |
| **G** | Naudojamas trumpesnis iš dviejų formatų: %E arba %f | **392.65** |
| **o** | Aštuntainis sveikas skaičius | **610** |
| **s** | Simbolių eilutė | **sample** |
| **u** | Beženklis sveikas skaičius | **7235** |
| **x** | Beženklis šešioliktainis skaičius | **7fa** |
| **X** | Beženklis šešioliktainis skaičius (didžiosios raidės) | **7FA** |
| **p** | Rodyklė | **B800:0000** |
| **%%** | Spausdinamas simbolis % | |

# *flag* reikšmės

| *flags* | Reikšmė |
|---|---|
| - | Lygiavimas pagal kairį kraštą. (Lygiavimas pagal dešinį yra default). |
| + | Spausdinamas skaičius su (+ arba -). (default – tik minusas išvedamas). |
| Tuščias tarpas | Jei argumentas yra teigiamas, tuščias tarpas įterpiamas prieš skaičių. |

**Pavyzdys**

printf("Mano batai buvo %-3d \n",2);  //Mano batai buvo 2

# *width* reikšmė

| ***width*** | **Reikšmė** |
|---|---|
| *Skaičius* | Minimalus simbolių skaičius, kuris bus išvestas. Jei reikšmė, kuri bus atspausdinta yra trumpesnė nei *skaičius*, tuomet likusios pozicijos užpildomos tarpais. Reikšmė niekad nėra apkarpoma, jei išvedama reikšmė didesnė (užima daugiau pozicijų)nei *skaičius*. |
| **0***skaičius* | Tas pats kaip viršuje, bet tarpai užpildyti **0** vietoj tarpų . |

**Pavyzdys**

printf("Mano batai buvo %03d \n",2);  // Mano batai buvo 002

# *precision* reikšmės

| ***precision*** | **Reikšmė** |
|---|---|
| ***.number*** | **d, i, o, u, x, X** tipams: *precision* nedaro įtakos. Jei reikšmė užima mažiau pozicijų nei *number* , tuomet likusios pozicijos užpildomos tarpais. Reikšmė niekad neapkarpoma, jei rezultatas užima daugiau pozicijų.<br><br>**e, E, f** tipams: *precision* apibrėžia minimalų skaičių po kablelio, kuris bus atspausdintas (Jei jis nenurodytas, pagal nutylėjimą **6**).<br><br>**g, G** tipams : nurodomas maksimalus reikšminių skaičių skaičius.<br><br>**s** tipui: maksimalus simbolių skaičius, kuris bus atspausdinas. (pagal nutylėjimą išvedama visa eilutė t.y. iki \0 simbolio). |

# Modifier reikšmė

| ***modifier*** | **Reikšmė** (t.y. kaip funkcija interpretuoja argumentus) |
|---|---|
| **h** | *argument* interpretuojamas kaips **short int** (integer types). |
| **l** | *argument* interpretuojamas kaip **long int** (interger types) or **double** (floating point types). |
| **L** | *argument* interpretuojamas kaip **long double** (floating point types). |

# printf() pavyzdys

```
#include <stdio.h>
int main ()
{ printf ("Characters: %c %c \n", 'a', 65);
  printf ("Decimals: %d %ld\n", 1977, 650000);
  printf ("Preceding with blanks: %10d \n", 1977);
  printf ("Preceding with zeros: %010d \n", 1977);
  printf ("Some different radixes: %d %x %o \n", 100, 100, 100);
  printf ("floats: %4.2f % +.0e  %E \n", 3.1416, 3.1416, 3.1416);
  printf ("Integer: %d \n", 10);
  printf ("%s \n", "A string");
  return 0;
}
```

Į ekraną

```
Characters: a A
Decimals: 1977 650000
Preceding with blanks:       1977
Preceding with zeros: 0000001977
Some different radixes: 100 64 144 0144
floats: 3.14 +3e+000 3.141600E+000
Integer: 10
A string
```