

2. Gyakorlat

Rövid elméleti összefoglaló

• Összehasonlító és logikai operátorok

- Relációjelek: <, <=, >, >=, == (egyenlő), != (nem egyenlő)
- Logikai operátorok: && (ÉS), || (VAGY), ! (NEM)

```
//MINTA2_01
#include <iostream>
using namespace std;
int main()
{
    int i = 1, j = 3, k = 12;
    if (i == 1 && j > 2)
        cout << "1. feltétel teljesult" << endl;
    else cout << "1. feltétel nem teljesult " << endl;
    if (i > 0 || k < 20)
        cout << "2. feltétel teljesult" << endl;
    else cout << "2. feltétel nem teljesult" << endl;
    if (i < -1 || j > 3 && k < 10)
        cout << "3. feltétel teljesult" << endl;
    else cout << "3. feltétel nem teljesult" << endl;
    cin.get();
    return 0;
}
```

A program futásának eredménye:

1. feltétel teljesult
2. feltétel teljesult
3. feltétel nem teljesult

• Léptető operátorok

- ++ (increment), a változó értékének eggyel való növelése: i++; ++i;
- -- (decrement), a változó értékének eggyel való csökkentése: j--; --j;

```
//MINTA2_02
#include <iostream>
using namespace std;
int main()
{
    int i = 1, j = 5, k, n, m;
    k = n = m = 0;
    cout << "i: " << i << " j: " << j << " k: "
        << k << " n: " << n << " m: " << m << endl;
    k += 2;
    cout << "i: " << i << " j: " << j << " k: "
        << k << " n: " << n << " m: " << m << endl;
    n = i++ + --j;
    cout << "i: " << i << " j: " << j << " k: "
        << k << " n: " << n << " m: " << m << endl;
    m = ++i + j--;
    cout << "i: " << i << " j: " << j << " k: "
        << k << " n: " << n << " m: " << m << endl;
    cin.get();
    return 0;
}
```

Megjegyzés: A prefixes alak (++i, --j) használata esetén a léptetés a kifejezés kiértékelése előtt megy végbe, és a változó az új értékkel vesz részt a kifejezés kiértékelésében. A postfixes alak (i++, j--) használata esetén a léptetés a kifejezés kiértékelése után megy végbe.

A program futásának eredménye:

i: 1 j: 5 k: 0 n: 0 m: 0
i: 1 j: 5 k: 2 n: 0 m: 0
i: 2 j: 4 k: 2 n: 5 m: 0
i: 3 j: 3 k: 2 n: 5 m: 7

• Bitműveletek (a felsorolás a precedencia sorrendjében történt):

- ~ 1-es komplement (bitenkénti tagadó),
- & bitenkénti ÉS,
- | bitenkénti VAGY,
- ^ bitenkénti kizáró VAGY.

```
//MINTA2_03
#include <iostream>
using namespace std;
int main()
{
    unsigned int bitek1 = 0163;
    unsigned int bitek2 = 0007;
    unsigned int bitek3 = 0142;
    unsigned int bitekEs, bitekVagy, bitekKizaroVagy, negalt;
    bitekEs = bitek1 & bitek2;
    bitekVagy = bitek1 | bitek3;
}
```

<pre> bitekKizaroVagy = bitek1 ^ bitek3; negalt = ~ bitek1; cout << oct << bitek1 << " & " << oct << bitek2 << " = " << oct << bitekEs << endl; cout << oct << bitek1 << " " << oct << bitek3 << " = " << oct << bitekVagy << endl; cout << oct << bitek1 << " ^ " << oct << bitek3 << " = " << oct << bitekKizaroVagy << endl; cout << " ~ " << oct << bitek1 << " = " << oct << negalt << endl; cin.get(); return 0; } </pre>	<p><i>A program futásának eredménye:</i></p> <pre> 163 & 7 = 3 163 142 = 163 163 ^ 142 = 21 ~ 163 = 37777777614 </pre> <p>Példaként az első feladat:</p> <pre> 0001 0110 0011 & 0000 0000 0111 ----- 0000 0000 0011 -> 3 </pre>
---	--

• Biteltoló műveletek

- \ll eltolás balra $a \ll n$ (2^n értékkel való szorzás): `szorzas = b << 1; //kettővel való szorzás`
- \gg eltolás jobbra $a \gg m$ (2^m értékkel való osztás) `osztas = c >> 1; // kettővel való osztás`

<pre> //MINTA2_04 #include <iostream> using namespace std; int main() { unsigned int bitek1 = 0004; unsigned int bitek2 = 0010; unsigned int szoroz2, oszt2; szoroz2 = bitek1 << 2; oszt2 = bitek2 >> 3; cout << oct << bitek1 << " << 2 = " << oct << szoroz2 << endl; cout << oct << bitek2 << " >> 3 = " << oct << oszt2 << endl; cin.get(); return 0; } </pre>	<p><i>A program futásának eredménye:</i></p> <pre> 4 << 2 = 20 10 >> 3 = 1 </pre> <p><i>Első feladat:</i> Léptetés előtt: 0000 0100 Léptetés után: 0001 0000 bináris 16 -> oktális 20</p> <p><i>Második feladat:</i> Léptetés előtt: 0000 1000 Léptetés után: 0000 0001 bináris 1 -> oktális 1</p>
--	--

Megjegyzés: A kiléptetett bitek elvesznek!

- A **sizeof** operátor megadja tetszőleges változó, típus, kifejezés típusának megfelelő bajtméretet.
 - **sizeof** változó
 - **sizeof** (típusnév)

<pre> //MINTA2_05 #include <iostream> using namespace std; int main() { int i = 1, ihossza, xhossza, xhossza; double x = 2.45; ihossza = sizeof i; xhossza = sizeof x; xthossza = sizeof(double); cout << "int merete : " << ihossza << " bajt" << endl; cout << "double merete: " << xhossza << " bajt" << endl; cout << "double merete: " << xthossza << " bajt" << endl; cin.get(); return 0; } </pre>	<p><i>A program futásának eredménye:</i></p> <pre> int merete : 4 bajt double merete: 8 bajt double merete: 8 bajt </pre>
---	---

• A C++ nyelv utasításai

- Tetszőleges kifejezés utasítás lesz, ha pontosvesszőt (;) helyezünk mögé.
- A kapcsos zárójelek { } a logikailag összefüggő definíciókat, deklarációkat és utasításokat egyetlen összetett utasításba, blokkba fogják össze.
- Feltételes operátor három operandussal rendelkezik: $kif_1 ? kif_2 : kif_3$

```

// a minimális adatot választja ki:
min = x < y ? x : y;

```

```
// szintén a minimális adatot választja ki:
```

```
x < y ? min = x : min = y;
```

- **if** (feltételes) utasítás:

if (kifejezés) { utasítások; }	Ha a <i>kifejezés</i> igaz (true , vagy a <i>kifejezés</i> értéke nem nulla), akkor a { } zárójelben lévő <i>utasításokat</i> hajtja végre.
if (kifejezés) <i>utasítás1</i> ; else <i>utasítás2</i> ;	Ha a <i>kifejezés</i> igaz (true , vagy a <i>kifejezés</i> értéke nem nulla), akkor az <i>utasítás1</i> -et hajtja végre. Ha a <i>kifejezés</i> nem igaz (false , vagy a <i>kifejezés</i> értéke nulla), akkor az <i>utasítás2</i> -öt hajtja végre.

<pre>//MINTA2_06 #include <iostream> using namespace std; int main() {int a = 14, b = 5, c; cout << "a: " << a << " b: " << b << endl; if (a > b) { c = a; a = b; b = c; // két változó tartalmát felcseréli } cout <<"Az if utasitas vegrehajtasa utan:" <<endl; cout << "a: " << a << " b: " << b; cin.get(); return 0; }</pre>	<p><i>A program futásának eredménye:</i></p> <p>a: 14 b: 5 Az if utasitas vegrehajtasa utan: a: 5 b: 14</p>
--	---

<pre>//MINTA2_07 #include <iostream> using namespace std; int main() { int n; cout << "Egesz szam : "; cin >> n; if (n % 2 == 0){ cout << "A szam paros!" << endl; } else { cout << "A szam paratlan!" << endl; } cin.get(); cin.get(); return 0; }</pre>	<p><i>A program futásának eredményei:</i></p> <p>Egesz szam : 12 A szam paros! Egesz szam : 13 A szam paratlan!</p>
--	--

- A **switch** utasítás (többirányú elágaztatás, a *kifejezés* csak sorszámozott típusú lehet).

```
switch (kifejezés)
{
    case konst_kifejezés:
        utasítások; break;
    ...
    default:
        utasítások;
}
```

- A **break** utasítással kiléphetünk a **switch** blokkból.

<pre>//MINTA2_08 #include <iostream> using namespace std; int main() { char c; cout << "Kerek egy karaktert: "; cin >> c; switch (c) { case 'a': cout << "karakter: " << c << endl; break; case 'b': cout << "karakter: " << c << endl; break; default: cout << "egyeb karakter" << endl; } cin.get(); cin.get(); }</pre>	<p><i>A MINTA2_08 program csak az a vagy b karakter beolvasásakor írja vissza a karaktert, minden további karakter egyéb karakternek számít.</i></p> <p><i>A program futásának eredményei:</i></p> <p>Kerek egy karaktert: + egyeb karakter Kerek egy karaktert: b karakter: b</p>
--	--

Feladatok

1. Olvassunk be egy egész számot 1 és 100 között! Ha értéke páros, osszuk el kettővel, azaz toljuk el jobbra (>>) egy helyiértékkal. Ha páratlan, akkor szorozzuk meg kettővel, azaz toljuk el balra (<<) egy helyiértékkal! Az eredményeket jelenítsük meg szövegesen! (GYAK2_1)

```
int szam, szorozva, osztva;
```

A program futásának eredményei:

```
Egesz szam: 12
leptetes jobbra (osztva 2-vel): 6

Egesz szam: 13
leptetes balra (szorozva 2-vel):26
```

2. Olvassunk be egy valós számot, ha értéke pozitív, vonjuk belőle gyököt, ha negatív, akkor emeljük négyzetre! (GYAK2_2)

```
double adat, eredmeny;
```

A program futásának eredményei:

```
Valos szam : 16
Negyzetgyok: 4

Valos szam : -8
Negyzete   : 64
```

3. Olvassunk be egy kor adatot, és írjuk az alábbi szövegek egyikét! (GYAK2_3)

Kor	Státusz
kor < 6	Nem iskolás
≤ 6 kor ≤ 14	Általános iskolás
< 14 kor ≤ 18	Gimnazista
< 18 kor ≤ 24	Egyetemista
< 25 kor ≤ 62	Dolgozó
kor > 62	Nyugdíjas

A program futásának eredménye:

```
Kor: 19
Egyetemista
```

4. Olvassunk be 1 és 6 közötti egy egész számot, szimulálva a kockadobást! Adjunk hibajelzést, ha rossz adatot adtunk meg! Írjuk ki a kockadobás értékét szövegesen! (GYAK2_4)

A program futásának eredményei:

```
Kockadobas (1-6) kozott: 0
Hibas adat!

Kockadobas (1-6) kozott: 4
Dobas erteke: NEGY

Kockadobas (1-6) kozott: 6
Dobas erteke: HAT
```

5. Definiáljuk az alábbi változókat, és írassuk ki a bájtban elfoglalt méretüket a **sizeof** operátor használatával! (GYAK2_5)

```
int i; long int j; float x; double y;
```