A C++ program nyelv operátorai

6.1. értékadó operátor =

A programozásban az értékadó operátor az egyenlőség jel. Ennek az operátornak a segítségével lehet az adott értéket eltárolni a változóban. Az operátor 2 operandusú. Az értékadás iránya csak jobbról balra lehetséges. Az operátor bal oldalán lévő változóban tároljuk a jobb oldalon lévő értéket.

- baloldalon csak változó állhat
- jobb oldalon lehet változó és a konstans is.

A matekban az x=2 és az 2=x ugyan azt jelenti, azonban a programozásban csak az x=2 alak lehetséges. A változó kezdeti érték adását hívjuk inicializációnak.

6.2. logikai operátorok

és vagy operátorok működését leginkább bool tábla segítségével érthetjük meg.

és operátor jele: &&

vagy operátor jele: ||

logikai tábla - boolean table

A	B	A & B	A || B	!A
0	0	0	0	1
1	0	0	1	0
0	1	0	1	1
1	1	1	1	0

A fenti táblázat tartalmazza, hogy A, B különböző értékeire az ÉS, VAGY operátorok milyen értéket adnak vissza.

pl.:

```
bool a = true;
bool b = false;
bool eredmeny = a && b;
cout << "a és b logikai kifejezés kiértékelésének eredménye: " << eredmeny << endl;
```

Az ÉS operátornál csak akkor kapunk igaz értéket ha mindkét operandus igaz. A fenti táblázat minden lehetséges esetet tartalmaz.

6.3 relációs operátor

A relációs operátorok (<,>) segítségével össze lehet hasonlítani 2 számot. A lent látható logikai kifejezést, ha kiértékeljük igaz vagy hamis logikai értéket kapunk. A kiértékelés eredményét eltároltam az eredmény változóban majd kiírtam a konzol ablakra.

```
int a = 10;
int b = 20;
bool eredmeny = a < b;
cout << "a < b logikai kifejezés kiértékelésének eredménye: " << eredmeny << endl;
```

6.4 maradék képező operátor

A maradék képező operátor segítségével az osztás után keletkező maradékot eltárolhatjuk egy változóban.

```
int a = 4;
int b = 2;
int maradek = a % b;
cout << "az osztás után keletkező maradék: " << maradek << endl;
```

6.5 ternary operátor ? :

kifejezés_1 ? kifejezés_2 : kifejezés_3

A feltételes operátornál az 1. logikai kifejezés kiértékelése lehet igaz vagy hamis. Ettől függően végrehajtódik a 2. vagy a 3. kifejezés. Ha a 1. kifejezés igaz, végrehajtódik a 2., ha hamis végrehajtódik a 3. kifejezés helyén lévő programkód.

```
#include <iostream>
using namespace std;
int main()
{
	bool test = false;
	int eredmeny = test ? 10 : 20;
	cout << eredmeny << endl;
	system("pause");
	return 0;
}
```

6.6 operátor precedencia

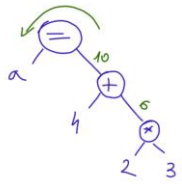
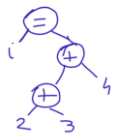
Az programkódban lévő operátorokat tartalmazó kifejezések kiértékelése csak a helyes sorrendben lehetséges, az operátorok precedenciájának megfelelően. Ezt hívjuk tömören operátor precedenciának. A kiértékelésnél két szempontot kell figyelembe venni: **erősség és kiértékelési irány.** A következő táblázatban az erősség fentről lefelé csökken. A kiértékelés iránya jobbról balra, vagy fordítva lehetséges.

erősség:	név:	jel:	irány:
1.	**postfix**	( ) [ ] -> . ++ --	→
2.	**unary prefix**	! ~ & sizeof ++ --	←
3.	**multiplicative**	* / %	→
4.	**additive**	+ -	→
5.	**shift**	<< >>	→
6.	**relational**	< > <= >=	→
7.	**equality**	== !=	→
8.	**and**	&&	→
9.	**or**	||	→
10.	**conditional**	? :	←
11.	**assignement**	= += -= *= /=	←
12.	**comma**	,	→

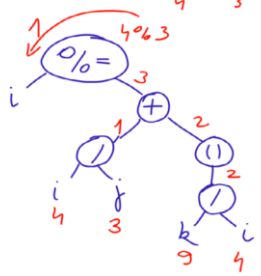
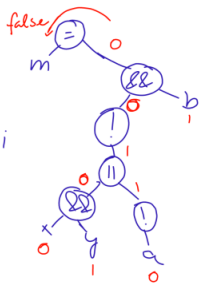
Az operátoroknál **++** jel állhat a változó előtt és után. Ha előtte áll pl.: **++i**, akkor **prefix** operátornak hívjuk, ha utána áll, pl.: **i++** akkor a **postfix** operátornak hívjuk. Az operátor lehet 2 vagy 1 operandusú. Egy operandusú pl.: **!a**, 2 operandusú pl.: az **a+b,** egyszerűen azért mert 2 változóra hat.

A kiértékelésnél segítséget nyújthat a precedencia fa használata.

1. feladat	2. feladat	3. feladat

int a = 4 + 2*3;
 
operátor precedencia erősségének sorrendje: * + =
A legerősebb precedenciával * rendelkezik, így vele kezdődik a precedencia fa alul.
	`int i = 3;`
	`int j, k;`
	`j = k = i;`
	Érték adó operátor kiértékelési iránya: ←
		int i = 2+3+4;
		 
		Az összeadó operátor kiértékelési iránya: balról jobbra →

4. feladat	5. feladat: logikai operátor

int j=3, k=9, i=4;
i %= i/j + (k/i);
i = i %3;
 
Alulra, a kezdéshez kerül a legerősebb precedencia.
	boolean x=false,
	y=true,
	a=false,
	b=true;
	boolean m = !(x&&y || !a) && b
	m: false
	 
	előforduló operátorok:
	() zárójel
	&& és
	|| vagy

	! tagadás operátorok erőssége: () ! és vagy - tagadással kezdünk