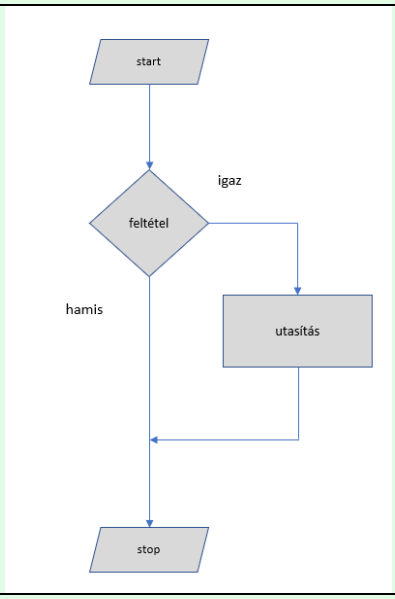
matek, fizika programozás könnyedén

A C++ nyelv vezérlő szerkezetei

1. if

Az 1. vezérlő szerkezet amit tanulmányozni fogunk az **if( )** -lesz. Ezt a vezérlő szerkezetet elágazásnak is szokás hívni.

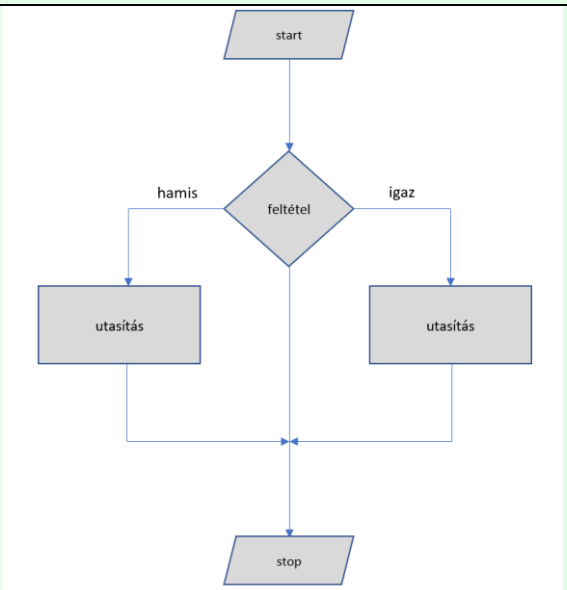



```
#include <iostream>
using namespace std;
int main()
{
	setlocale( LC_ALL, "hun" );
	int a = 10;
	if ( a > 5 )
	{
		cout << "Az a > 5!" << endl;
	}
	system("pause");
	return 0;
}
```

A végrehajtási sorrend amikor megérkezik az **if** vezérlő szerkezethez, a belépési feltétel kiértékelődik és annak igazság tartalma szerint tovább halad az igaz vagy a hamis ág felé. A programvégrehajtás így elágazott azaz 2 irányú lett. Az $\mathbf{a} > \mathbf{5}$ logikai kifejezés mint feltétel segítségével irányítani lehet a program lefolyását, éppen ezért hívják elágazásnak. Az $\mathbf{a} > \mathbf{5}$ szöveg csak akkor íródik ki ha az **a** változó tényleg nagyobb mint 5.

2. if – else

Az **if - else** a korábbiaktól csak annyiban különbözik, hogy az elágazás hamis ágában is van utasítás. Ha az **if** belépési feltétele hamisnak értékelődik ki akkor is van kiírt üzenet.

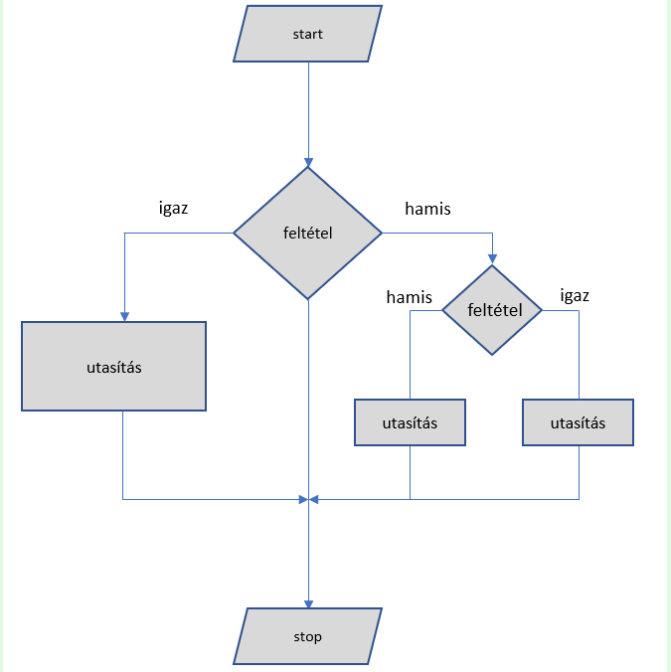



```
#include <iostream>
using namespace std;
int main()
{
	setlocale(LC_ALL, "hun");
	int a = 10;
	if ( a > 5 )
	{
		cout << "Az a > 5!" << endl;
	}
	else
	{
		cout << "Az a nem > mint 5!" << endl;
	}
	system("pause");
	return 0;
}
```

3. egymásba ágyazott elágazások

Egy elágazás igaz és hamis ágába is további elágazások illeszthetőek be. A lenti példában a hamis ágban egy újabb **if - else** szerkezet van beépítve. Ennek segítségével azt lehet elérni, hogy a belső **if** belépési feltétele csak akkor értékelődik ki ha a **külső** szerkezet-é hamis lett. Ezzel a módszerrel egymást kizáró eseményeket tudunk végrehajtani. Erre példa egy olyan eset amikor egy szám 3-mal való oszthatóságát csak akkor vizsgáljuk meg ha a 4-gyel való nem teljesül.

if - else - if



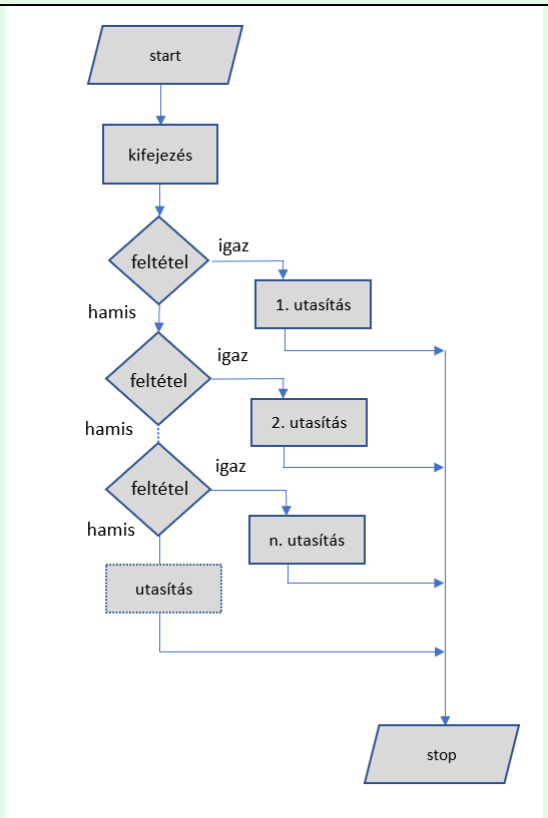

```
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL, "hun");
    int a = 10;
    if ( a > 5 )
    {
            cout << "Az a > 5!" << endl;
    }
    else
    {
        if ( ... )
        {
                cout << "........." << endl;
        }
        else
        {
                cout << "........." << endl;
        }
    }
    system("pause");
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
    setlocale(LC_ALL, "hun");
    int a = 10;
    if ( a > 5 )
    {
        cout << "Az a > 5!" << endl;
    }
    else if(...)
    {
        cout << "........." << endl;
    }
    else
    {
        cout << "........." << endl;
    }
    system("pause");
    return 0;
}
```

Az **if - else -if** lényegében egy rövidítés ami arra utal, hogy az else ágban további **if - else** van beépítve. Ezt az eljárást tovább lehet folytatni és a belső **if else**, **else** ágába újabb **if - else** szerkezet építhető be.

4. switch - case

A többirányú elágazás másnéven összetett elágazás több végrehajtási ággal rendelkezik, és egyetlen kifejezés kerül kiértékelésre, melynek eredménye alapján kerül az egyes ágakra a vezérlés. Ha végrehajtódott az adott ág, akkor a **break** utasítás miatt a vezérlési szerkezet végére kerül a végrehajtás.




```cpp
#include <iostream>
using namespace std;

int main()
{
        setlocale(LC_ALL, "hun");
        int a = 0;
        int b = 0;
        cout << "Kérem adjon meg két számot!" << endl;
        cin >> a;
        cin >> b;
        int menu = 0;
        cout << "Kérem válasszon menüből!" << endl;
        cout << "1. + " << endl;
        cout << "2. - " << endl;
        cout << "3. * " << endl;
        cout << "4. / " << endl;
        cin >> menu;
        switch (menu)
        {
        case 1:
                cout << "+ : "<< a + b << endl;
                break;
        case 2:
                cout << "- : " << a - b << endl;
                break;
        case 3:
                cout << " * : " << a * b << endl;
                break;
        case 4:
                cout << "/ : "<< (float)a / (float)b << endl;
                break;
        default:
                cout << "Hiba!" << endl;
        }
        system("pause");
        return 0;
}
```

A cin >> utasítás segítségével (consol in) bekérhetünk egy számot a felhasználótól.

```
int menu = 0;
cin >> menu;
```

A default ágba akkor kerül a végrehajtás ha egyik esetnek, „case” -nek megfelelő ág sem jön létre a belépési feltételben.

```
#include <iostream>
using namespace std;
int main()
{
	setlocale(LC_ALL, "hun");
	int a,b = 0;
	cout << "Kérem adjon meg két számot!" << endl;
	cin >> a;
	cin >> b;
	int menu = 0;
	cout << "Kérem válasszon menüből!" << endl;
	cout << "1. + " << endl;
	cout << "2. - " << endl;
	cout << "3. * " << endl;
	cout << "4. / " << endl;
	cin >> menu;
	switch (menu)
	{
	case 1:
		cout << "+ : "<< a + b << endl;
	case 2:
		cout << "- : " << a - b << endl;
		break;
	case 3:
		cout << "* : " << a * b << endl;
		break;
	case 4:
		cout << "/ : "<< (float)a / (float)b << endl;
		break;
	default:
		cout << "Hiba!" << endl;
	}
	system("pause");
	return 0;
}
```

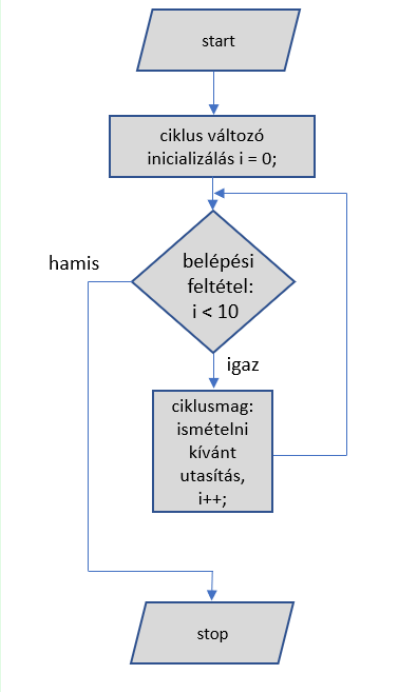
Az itt látható kódban a case 1 ágban nincs **break** utasítás. Ez nem túl célszerű mert ha a **case 1** kerül végrehajtásra akkor a **case 2** is végrehajtódik pedig azt nem választottuk ki.

5. ciklus

A ciklus segítségével a blokkban lévő kód többszöri végrehajtására van lehetőség. A folyamatot megint egy belépési feltétel irányítja. A feltétel logikai típusú, melynek tesztelése 2 helyen is előfordulhat.

elöl tesztelő ciklus: while, for

Az elöltesztelő ciklus először megvizsgálja, hogy a feltétel teljesül-e. Ha igen, akkor végrehajtja a ciklusmagot, majd a folyamat az újból elindul. Ha nem, akkor a program a ciklus utáni ponton folytatódik tovább, azaz a **ciklusmag** kimarad.

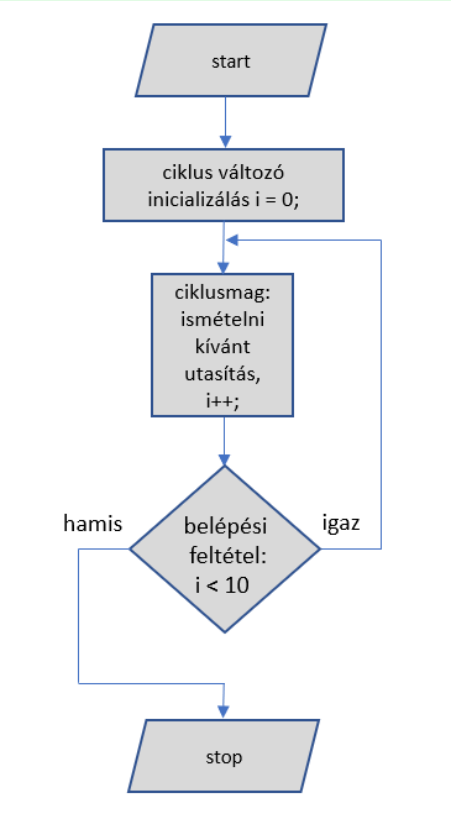



```
#include <iostream>
using namespace std;
int main( )
{
		setlocale( LC_ALL, "hun" );
		int size = 10;
		int i = 0;
		while ( i < size )
		{
			cout << "i : " << i << endl;
			i++;
		}
		system( "pause");
		return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
		setlocale(LC_ALL, "hun");
		int size=10;
		for ( int i = 0; i < size; i++ )
		{
			cout << "i : " << i << endl;
		}
		system("pause");
		return 0;
}
```

hátul tesztelő ciklus: do - while

A hátultesztelő ciklus először végrehajtja a ciklusmagot utána megvizsgálja, hogy a feltétel teljesül-e.




```
#include <iostream>
using namespace std;
int main( )
{
		setlocale( LC_ALL, "hun" );
		int size = 10;
		int i = 0;
		do
		{
			cout << "i : " << i << endl;
			i++;
		}
		while ( i < size );
		system( "pause" );
		return 0;
}
```