Programozás II.

3. gyakorlat

C++ újdonságok

- Objektum-orientáltság
- Névterek
  - A forrásfájlokat névterekbe rendezhetjük
  - Hierarchikus tárolási mód (névtérben lehetnek újabb névterek)
  - Független a fájlok fizikai helyétől
  - using paranccsal kijelölhető, mely névtereket használjuk.
    - using namespace std; - az alapvető i/o műveletekhez
    - std::cout

C++ újdonságok

- Egysoros komment (C-ben csak /* */)

  // Komment egyenértékű: /* Komment */

- For-beli ciklusváltozó létrehozása és inicializálása a for cikluson belül

```
for ( int i=0; i < 10; i ++ ) {
    //blokk
}
```

- bool típus (C-ben false = 0, true = minden más)
  - bool igaz = true; //értéke: 1
  - bool hamis = false; //értéke: 0

I/O műveletek

- **#include <iostream>**
  - C-s megfelelője az stdio.h
  - Alapvető input/output műveleteket tartalmazza
- **cout, cerr – Alapértelmezett (hiba)kimenet**
  - stream (bájtok sorozata), melyeket << operátorral „köthetünk össze”
  - cout << ”Hello World” << ”!”;
- **cin – Alapértelmezett bemenet**
  - Innen olvas, változó típusának megfelelő értéket
  - >> operátorral használható, akár több beolvasásra is:
    int x, y;
    cin >> x >> y;

Default paraméterek

- Függvények paramétereinek **alapértelmezett értéket** is adhatunk, de ezeknek a paramétereknek **a paraméterlista végén** kell lenniük!
- Ha kevesebb paraméterrel hívjuk meg a függvényt, az alapértelmezettek megkapják az előre beállított értéket.

```
int nagyobb (int a, int b = 0) {
    return (a > b) a : b;
}

int x = nagyobb (2, 3);          // x = 3
int y = nagyobb (2);             // y = 2, ugyanaz, mint a nagyobb(2, 0)
```

Osztály létrehozása

- **class** kulcsszóval definiáljuk az osztályt, melyet az osztály **neve** követ
- { } – blokkban definiáljuk az osztály adattagjait, függvényeit, stb…
- **Fontos, hogy az osztályt egy ; jel zárja!**
  - Példa:

```
class SimpleClass {
        //…
};
```

Láthatóság

- Az osztályok adattagjainak és függvényeinek kívülről való elérhetőségét szabályozhatjuk.
  - **private**: az **alapértelmezett** láthatóság. Csak osztályon belül hivatkozhatunk rá.
  - **protected**: osztályon belül, és származtatott osztályokból érhető el
  - **public**: bárhonnan elérhető
- Láthatóság megadása:
  - megadjuk a láthatóságot, majd egy kettőspont (:) karaktert teszünk. Minden tag, amit ez után definiálunk ezt a láthatóságot fogja megkapni.

Példa

```
class SimpleClass {
    private:
        int a;
        void foo(){ //... }
    public:
        int b;
        int c;
    protected:
        int d;
        int foo2(int b) { //... }
};
```