```
typedef int myint;
…
int fv(int);
void fv(myint); //OK!
```

→C++ objektum orientált elemei

1.) objektum orientáltság bevezetése

egységbezárás alapelve (encapsulation)
hasonló tulajdonságok összegzése (class Könyv) → objektumok (egy bizonyos könyv)

adatrejtés alapelve
az objektumnak csak egy része látható a külvilág számára (az objektum tud magára vigyázni)

öröklés
„is-a" relationship … (cow is an animal) = cow → állat (→általánosítás ill. ← specializáció) [a speciális örökli az általános tulajdonságait + műveleteit]

behelyettesíthetőség
ahol az általános szerepel ott a speciális is

típustámogatás
osztályok úgy működhetnek mint a beépített típusok

2.) Az egységbezárás alapelve (class)

bevezető

```
const unsigned int MAX-x = 1023;
const unsigned int MAX-y = 767;

struct Point
{
        unsigned int x;
        unsigned int y;
};

…

int setX(Point * point, unsigned int value)
{
        if(value <= MAX_x)
        {
                point->x = value;
                return 0;
        }
        return -1;
```

```
}

int Point_setY(Point * point, unsigned int value)
{
	if(value <= MAX_y)
	{
		point->y = value;
		return 0;
	}
	return -1;
}

int main()
{
	Point p1;
	Point_setX(&p1, 100);
	Point_setY(&p1, 200);
}
```

//C++ban lehet a struktúrának függvény elemei is (az előző feladat)

```
struct Point
{
	unsigned int x, y;
	int setX(unsigned int value)
	{
		if(value <= MAX_x)
		{
			x = value;
			return 0;
		}

		return -1;
	}

	int setY(unsigned int value)
	{
		if(value <= MAX_y)
		{
			y = value;
			return 0;
		}

		return -1;
	}

}
```

```
int main()
{
	Point p1;
	p1.setX(100);
	p1.setY(200);
}
```

//ezt is lehet csinálni

.h fájlba:

```
struct Point
{
	unsigned int x, y;
	int setX(insigned int value);
	int setX(insigned int value);
}
```

.cpp-be

```
int Point :: setX(unsigned int value)
{
	…
}
```

:: scope (hatókör) operátor

Point_ setX(&p1, 25);

p1.setX(25);

t.f.h.

```
struct Point
{
	unsigned int x, y;
	int setX(insigned int value);
	int setX(insigned int y);
}
```

akkor:

```
int Point :: sety(unsigned int y)
{
	…
	this->y = y; (láthatatlan pointerre való hivatkozás)
}
```

tagváltozók

(struktúrán belüli változók)

HIBA(C-ben OK volt)

fwrite(&p1, sizeof(p1),1,fp);

Adatrejtés

Point p1;
p1.x = MAX_X + 3; (így az x-et inkonzisztens állapotba hoztuk) //hiba

---

```
struct Point
{
        private:
        unsigned int x, y;
        public:
        int setX(unsigned int);
        int setY(unsigned int);
        //kell írni függyvényeket amikkel x, y értékét visszakérhetjük
        unsigned int getX() return x;
        unsigned int getX() return y;
};
```

p1.x → ez itt már hiba!! nem hivatkozhatunk rá!
p1.getX(); //OLLÉ

3.) **Class**

→ **alapértelmezettként private a class elérése!** (struct public, C fordítás miatt)

```
class Point
{
        …
};
```

Point p1; //p1 objektum

4.) **Konstruktoruk/destrukturok**

Point p1; //azt akarjuk, hogy kezdetben is legyen értéke a class-on belül

```
class Point
{
        unsigned int x,y;
        public:
        …
        Point(){x =0; y=0;}
        Point(unsigned int x, unsigned int y){this->x = x; this->y = y;} //vizsgálat!
}

Point p2;       //0,0
p2.setX(100);
p2.setY(300);   //ez pocsékolás(?)

Point p3(100, 200);

Point p4();  //HIBA!! 0 paraméterű konstruktort úgy kell hívni mint p2-t

int setY(unsigned x, unsigned y)
{
        Point(x, y);        //ÓRIÁSI HIBA!!!!
}
```