Membri pointeri

- Realizează extensii ale clasei în heap
- Dacă sunt vectori numerici (nu char*) atunci implică definirea unui câmp suplimentar ce salvează numărul de elemente
- Atunci când o clasă are un membru pointer, este obligatorie definirea explicită a următoarelor metode (cunoscută și ca „regula celor 3”):
  - destructor
  - constructor de copiere
  - operator de atribuire (operator egal)

Cazuri speciale în clasele cu membri pointeri

- Orice copiere a membrului pointer se va face într-o zonă de memorie nouă
- Copiere unei adrese cu = va crea o copie superficială, ceea ce va face ca ambele entități să partajeze aceeași zonă de memorie
- Getter-ul pentru membrul pointer va returna o copie a vectorului din clasă
- Setter-ul pentru membrul pointer de tip vector numeric va primi 2 parametri (noul vector și noul număr de elemente)

Exemplu

```
class TelefonMobil
{
//...
private:
	string producator;
	int* durataZilnicaUtilizare;
	int nrZile;
//...
};
```

Getter pentru membrul pointer

```
class TelefonMobil
{
//...
private:
        string producator;
        int* durataZilnicaUtilizare;
        int nrZile;
public:
        int* getDurataZilnicaUtilizare()
        {
                if(durataZilnicaUtilizare == NULL || nrZile <= 0) return NULL;
                int* copie = new int[nrZile];
                for(int i = 0; i < nrZile; i++)
                {
                        copie[i] = durataZilnicaUtilizare[i];
                }
                return copie;
        }
//...
};
```

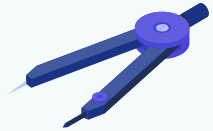
Getter pentru membrul pointer (vers 2)

```
class TelefonMobil
{
//...
private:
        string producator;
        int* durataZilnicaUtilizare;
        int nrZile;
public:
        int getDurataZilnicaUtilizare(int index)
        {
                if(index >= 0 && index < nrZile)
                {
                        return durataZilnicaUtilizare[index];
                }
                else return -1;
        }
//...
};
```

Setter pentru membrul pointer

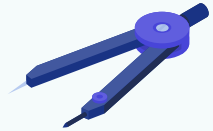
```
class TelefonMobil
{
//...
private:
        string producator;
        int* durataZilnicaUtilizare;
        int nrZile;
public:
        void setDurataZilnicaUtilizare(int* durate, int nr)
        {
                if(durate != NULL && nr > 0)
                {
                        if(durataZilnicaUtilizare != NULL) delete[] durataZilnicaUtilizare;
                        nrZile = nr;
                        durataZilnicaUtilizare = new int[nr];
                        for(int i = 0; i < nr; i++)
                                durataZilnicaUtilizare[i] = durate[i];
                }
        }
//...
};
```

Operatorul de atribuire (operator =)



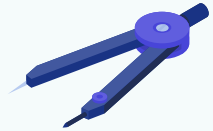
- Apelat automat atunci când se copiază informații dintr-un obiect existent într-un alt obiect existent (spre deosebire de constructorul de copiere ce copiază informații dintr-un obiect existent într-un obiect nou)
- Poate returna void (situație în care nu este permis apelul în cascadă) sau adresa obiectului tocmai creat (ce permite apelul în cascadă)
- La fel ca și constructorul de copiere primește drept parametru obiectul din care se face copierea

Definire explicită operator=



```
class TelefonMobil
{
public:
//...
	TelefonMobil& operator=(const TelefonMobil& telefon)
	{
		producator = telefon.producator;
		model = telefon.model;
		nivelBaterie = telefon.nivelBaterie;
		//...
		return *this;
	}
//...
};
```

Utilizare



```
int main()
{
	TelefonMobil telefonPersonal("Samsung", "S20");
	TelefonMobil telefonDeServiciu(telefonPersonal);
	telefonDeServiciu = telefonPersonal;
	//nu se apeleaza operator= ci constructorul de copiere
	TelefonMobil telefon2 = telefonPersonal;
	//apel in cascada, permis doar daca operator= NU returneaza void
	telefon2 = telefonPersonal = telefonDeServiciu;
}
```

Compunere claselor

- Pot exista situații când o clasă va avea membri de tipul unei alte clase
- Acest lucru poartă denumirea de compunere și este primul tip de relație dintre clase
- Compunerea claselor răspunde la întrebarea „has a?” („are?” / „are un?” / „are o?”)
- Pot fi modelate atât relații de top 1 la 1 cât și 1 la mulți

Compunerea claselor (relație 1:1)

```
class Baterie
{
     //...
};
class TelefonMobil
{
     string producator;
     string model;
     Baterie baterie;
     //...
};
```

Compunerea claselor (relație 1:M)

```
class CartelaSim
{
     //...
};
class TelefonMobil
{
     string producator;
     string model;
     CartelaSim cartele[2];
     //...
};
```