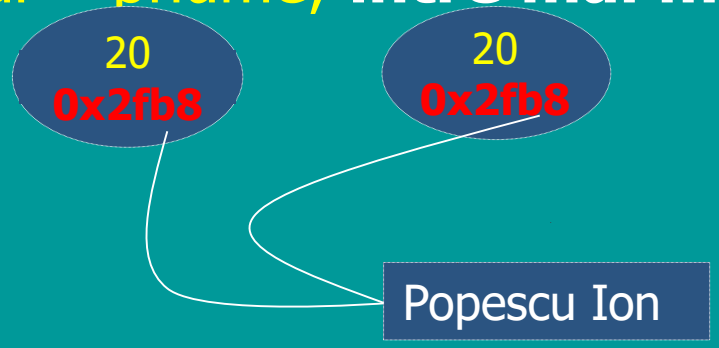
Copy constructor

- **Rolul constructorului de copiere:** construirea de obiecte din alte obiecte: Pers (Pers &src)
- **Utilizare:**
  1. initializare: Pers p1("Adam Dan", 2500), p2(p1), p3=p1;
  2. apel de functie prin valoare: void f1(Pers p) { /* */}
  3. return prin valoare: Pers f1() {/* */ return p1;}

- **Inconveniente copy cons pus de compilator**
  1. **copiaza tot:** ContBancar c2(c1); duplicare sold**?**
  2. **partajeaza zona pointată de** char * pnume; **intre mai multe obiecte**



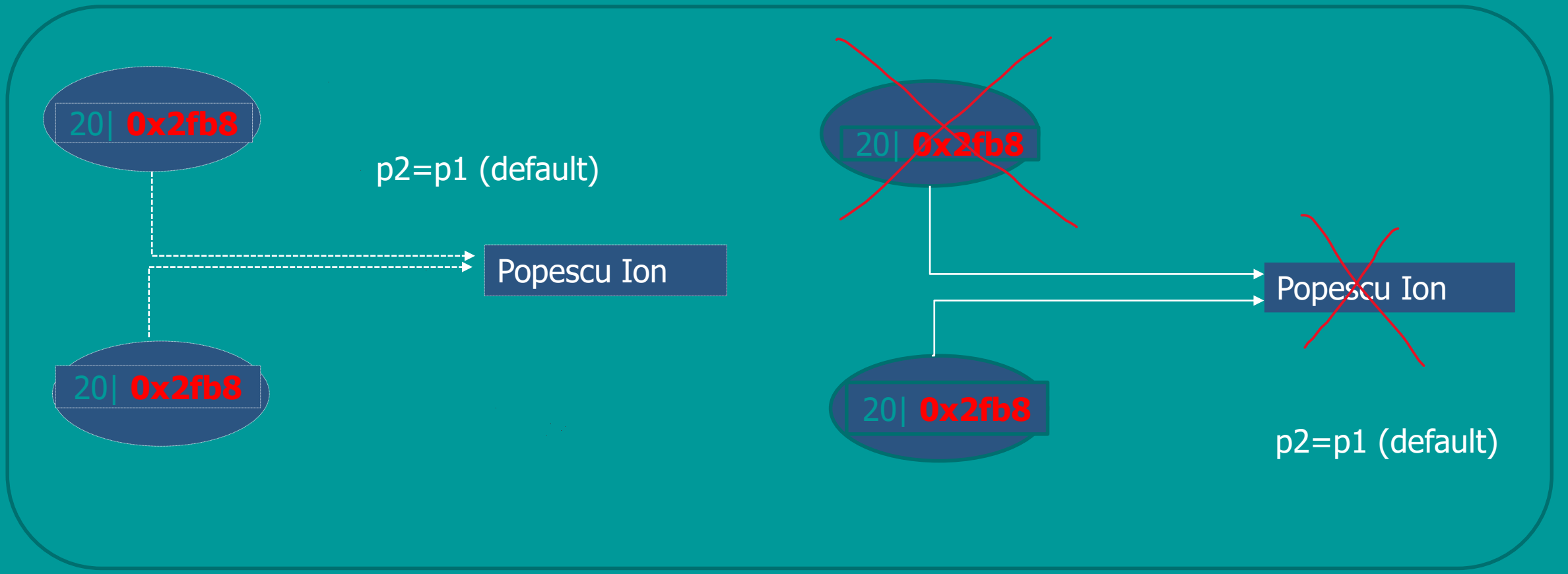
Membri de tip pointer
- obiecte cu extensii in memorie dinamică

- **necesitate**
  - **gestiune eficientă a memoriei (**char *pnume alocă strictul necesar**)**
  - **pointer la Server (** static Index *idx; obiectele retin unde sunt indexate)
  - structuri de clase implementate prin **pointeri la obiecte membre (**Auto * a;**)**
  - **structuri autoreferite (** node *next**)**

- **elemente obligatorii in clasa**
  1. **constructori de clasa** – pentru a face alocarea zonei dinamice
  2. **constructor de copiere** – pentru a face alocarea zonei dinamice
  3. **destructor** – pentru a dezaloca zona dinamică
  4. **operator=** – pentru a dezaloca vechea zonă și realoca alta, plus copiere

class Pers { int varsta; char* pnume; /* ... */ };

***operator=* pus implicit de compilator (default)**



```
Pers& operator=(Pers & src)
    {
        if (this != &src)
        {
            if (pnume != nullptr) delete[] pnume;
            pnume = new char[strlen(src.pnume) + 1];
            strcpy(pnume, src.pnume);
            marca = src.marca; salariu = src.salariu;
        }
        else cerr << "\nEroare auto-asignare";
        return *this;
    }
```

class Pers { int varsta; char* pnume; /* ... */ };

operator= **scris de programator (owner)**