evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

DUM 19 téma: Objektově orientované programování

ze sady: 2 **tematický okruh sady:** Vyšší programovací jazyky
ze šablony: 10 – Algoritmizace a programování **určeno pro:** 1. a 2. ročník
vzdělávací obor: 18-20-M/01 Informační technologie
26-41-M/01 Elektrotechnika - Elektronické počítačové systémy
vzdělávací oblast: odborné vzdělávání
metodický list/anotace: VY_32_INOVACE_10219ml.pdf
pomocné soubory: uk1.cpp, uk2.cpp, uk3.cpp, uk4.cpp, uk5.cpp, uk6.cpp, objekty.cpp

Objektově orientované programování je metoda programování, která je v poslední době hodně populární. Některé jazyky považují objektové programování za základ (Java, C#) do jiných programovacích jazyků se tato metoda dodala jako rozšíření (C se doplnilo o objekty a vzniklo C++).

Objektově lze tedy programovat pouze v jazyce C++, proto od této doby je nutné používat koncovku souboru zdrojového kódu ***.cpp***

Programování se stále vyvíjí a existovala a stále existuje celá řada metod. Např. strukturované programování (na kterém jsme se doposud učily), modulární programování nebo událostmi řízené programování (s využitím objektů) a dále pak celá řada neprocedurálních metod programování jako je logické či funkcionální programování.

I. Princip

Nejprve se pokusíme vysvětlit, kde se objektově orientované programování (OOP) vzalo a jak že lze objekty chápat jako rozšíření datových struktur.

a. Základní myšlenky

Svět se skládá z různých objektů (auta, domy…). Některé jsou speciální případy ostatních (škodovka je speciální případ auta) a mají tudíž některé společné vlastnosti. Jednotlivé objekty se mohou vzájemně ovlivňovat.

Původně se tento přístup využíval v počítačových simulacích ve vědeckém výzkumu. Postupně se tento přístup využil v řadě programů (např. vektorový grafický editor).

Velkou výhodou je možnost rozdělit program na jednotlivé části a definovat pouze rozhraní (vzájemná interakce jednotlivých částí). To umožňuje zejména:

1. Práce více programátorů na jednom programu.
2. Mohu snadno změnit (upgradovat) jednotlivé části, pokud zachovám rozhraní.

b. Technické/syntaktické provedení

Struktura – data, která spolu souvisí, jsme zapouzdřili do jednoho datového typu.

Objekt – data a zároveň funkce, které souvisí s těmito daty, zapouzdříme do jednoho datového typu = třída.

c. Třída X objekt

Třída – datový typ, který obsahuje data a zároveň funkce, které spolu souvisí (např. třída kružnice: data – střed, poloměr, funkce – posunutí, zvětšení).

Objekt – instance třídy (podobně jako `int číslo; cislo` je instance datového typu `int`). To znamená „jeden konkrétní objekt“ (např. `k1…S=[2,1], r=5`).

II. Objekt, třída, metoda

a. Definice třídy a vytvoření objektu

```
class student{
  public:
    char jmeno[20];
    char prijmeni[20];
    int rc;
};
```

```
int main(){…
 student a;
 strcpy(a.jmeno,"Karel");
 strcpy(a.prijmeni,"Novak");
 a.rc=901025;
… }
```

b. Metody

```
class student{…
 void vypis(){
  printf("%s %s
    %d",jmeno,prijmeni,rc);}
};
```

```
int main(){…
    a.vypis();
 …}
```

c. Konstruktor

Speciální funkce, která se volá při vytvoření objektu. Funkce má stejný název jako třída a nic nevrací.

```
class student{...
 student(char *a,char *b,int
r){
   strcpy(jmeno,a);
   strcpy(prijmeni,b);
   rc=r;
  }
};
```

```
int main(){…
  student
a("Karel","Novak",901025);
}
```

III. Základní vlastnosti OOP

Objektové programování by se neobešlo bez třech základních vlastností, které si nyní popíšeme. Jedná se o zapouzdření, dědičnost a polymorfismus.

a. Zapouzdření

Umožňuje rozdělit data a metody objektu na dvě skupiny: PUBLIC, PRIVATE.

1. PUBLIC – jsou viditelné z vnějšku. Data lze číst a ukládat do nich data, metody lze volat z jiných objektů nebo z programu.
2. PRIVATE – nejsou viditelné z vnějšku. Nelze číst ani měnit data nebo volat metody z jiných objektů nebo programu. Jsou dostupné pouze z objektu, ve kterém jsou vytvořeny.

Výhody: Pomocí PUBLIC dat a metod definuji rozhraní objektu (co mohou využívat ostatní). PRIVATE data a metody slouží k ochraně – nikdo jiný nezmění obsah těchto proměnných.

Syntaxe: V těle definice třídy pomocí klíčových slov public: , private:

```
class student{
  public:
    char jmeno[20];...
  private:
    int den,mesic,rok;...
};
```

b. Dědičnost

Pomocí dědičnosti mohu vytvářet objekty odvozené z jiných objektů. Některé objekty jsou totiž speciálními případy jiných objektů a lze tudíž využít některá data nebo metody, které jsme již jednou vytvářeli. Př:

1. Student
   1.1. Středoškolák
      1.1.1. Student gymnázia
      1.1.2. Student odborné školy
      1.1.3. Student učiliště
   1.2. Vysokoškolák

Vidíme, že každý středoškolák je student.
Každý student odborné školy je středoškolák a tedy i student.
Určitě bude platit, že jak středoškolák, vysokoškolák nebo student kterékoli střední školy bude mít jméno příjmení, datum narození. Není nutné, pro každou třídu tyto proměnné definovat, ale můžeme je ***zdědit*** od předchozí třídy ***student***.

Říkáme, že student je ***předek*** středoškoláka a že středoškolák je ***potomek*** studenta.

Středoškolák , jako potomek, zdědí všechny vlastnosti předka a dále mu můžeme definovat další metody a data, který předek nemá.

```
class vysokoskolak:public student{
  public:
    char univerzita[20];
    char fakulta[20];
    void vypis(){
      printf("%s %s %d %s
%s\n",jmeno,prijmeni,rc,univerzita,fakulta);
    }
 };
```

c. Polymorfismus

Polymorfismus neboli mnohoznačnost. Stejné věci (rozumějme metody) se u různých objektů dělají jinak (např. v grafice každý objekt můžeme zvětšit, ale je jasné, že čtverec se bude zvětšovat jiným způsobem než kružnice).

Mám stejnou metodu u všech tříd v hierarchii předek – potomek a musím zajistit, aby se mi pro každý objekt zavolala správná metoda.

Syntakticky: zjednodušeně lze říci, že před každou metodou napíšeme klíčové slovíčko ***virtual*** (včetně předka, ve které se tato metoda nedá spustit a nedává smysl).

```
class student{
 public:
  ...
   virtual void ohodnot(){};
};
class epecko: public student{
  public:
    virtual void ohodnot(){
      hodnoceni=1;
    }
};

class itecko: public student{
  public:
    virtual void ohodnot(){
      hodnoceni=5;
    }
};
```