Uniwersytet Zielonogórski
Wydział Informatyki, Elektrotechniki i Telekomunikacji
Instytut Sterowania i Systemów Informatycznych
Informatyka stacjonarne-dzienne pierwszego stopnia z tyt. inżyniera

Laboratorium Programowania Obiektowego

Polimorfizm i funkcje wirtualne

Ćwiczenia

1. Niech klasy X i Y będą zdefiniowane

```
class X {
  public:
    virtual f(void);
    f1(void);
} x, *px;
class Y: public X {
  public:
    virtual f(void);
    f1(void);
} y, *py;
```

Dany jest fragment programu:

```
px=&y;
px->f();
px->f1();
```

Które funkcje są wywoływane w powyższym fragmencie programu.

2. Dla następującego schematu dziedziczenia: A⟶B⟶C

   a) zdefiniować klasy przy założeniu, że klasa A ma być klasą abstrakcyjną,

   b) wyposażyć wszystkie klasy w wirtualną metodę *nazwa* zwracającą nazwę klasy,

   c) dodać destruktory wirtualne,

   d) dodać funkcję globalną zaprzyjaźnioną z klasą A, tak aby mogła wykonać to samo co funkcje z pktu b).

3. Napisać definicję klasy z trzema przeładowanymi funkcjami wirtualnymi. Stworzyć nową klasę dziedzicząca po pierwszej, z tylko jedną przeładowaną funkcją wirtualną. Utworzyć obiekt klasy pochodnej i odpowiedzieć na następujące pytania:

   a) Czy można wywołać wszystkie trzy bazowe funkcje poprzez obiekt klasy pochodnej i ewentualnie w jaki sposób?

   b) Czy po zrzutowaniu adresu klasy pochodnej na adres klasy bazowej można wywołać wszystkie funkcje?

   c) Czy po usunięciu definicji funkcji w klasie pochodnej można wykonać podpunkt a)?

4. Poprawić ewidentne błędy w programie, a następnie przeanalizować jego działanie.

```
#include<iostream.h>
class X{
public:
static int x1;
void virtual fun1(){cout<<"funkcja fun1 z klasy X";}
static void virtual fun2()=0;
```

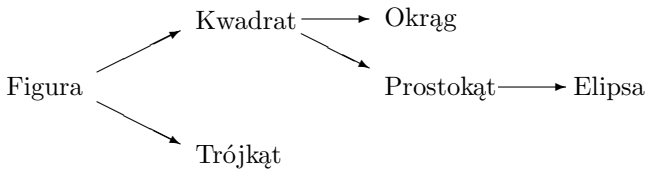
```
};
class Y: public X{
public:
void virtual fun1(){cout<<"funkcja fun1 z klasy Y";}
void fun2(){cout<<"funkcja fun2 z klasy Y";}
};
class Z: public Y{
public:
void virtual fun1(){cout<<"funkcja fun1 z klasy Z";}
void virtual fun2()=0 {cout<<"funkcja fun2 z klasy Z";}
};

void main(){
X ob1;Y ob2; Z ob3;
ob2.fun1;ob2.fun2;
X *wsk=&ob2;
wsk->fun1();wsk->fun2();wsk->X::fun1();wsk->X::fun2();
wsk=&ob3;
wsk->fun1();wsk->fun2();wsk->X::fun1();wsk->Z::fun2();
}
```

5. W ćwiczeniu 1 zastanowić się jak zrealizować zadanie konstruktora wirtualnego.

Zadania do samodzielnego wykonania

1. (**Obowiązkowe!**) Zrealizować poniższy schemat dziedziczenia:



   Każda z klas powinna posiadać funkcje wirtualne $rysuj(), przesun()$ i $obrot()$. Dokonać wirtualnego wywołania tych funkcji dla różnego typu obiektów.

2. (**Obowiązkowe – koszt polimorfizmu!**) Napisać klasę zawierającą dwie metody wykonujące to samo zadanie, jedną w wersji wirtualnej, a drugą niewirtualnej. Odziedziczyć wspomnianą klasę w nowej klasie potomnej, a następnie porównać czasy wywołania każdej z funkcji (posłużyć się np. funkcją *clock*() z biblioteki *time.h*).

3. Dana jest klasa bazowa zawierająca pewne składniki danych oraz klasa pochodna zawierająca dodatkowe składniki. Napisać globalną funkcję pobierającą obiekt klasy bazowej przez wartość i zwracającą jego rozmiar (funkcja $sizeof()$). W programie głównym zadeklarować obiekt klasy pochodnej, wypisać jego rozmiar, a następnie wywołać funkcję globalną. Wyjaśnić powstały efekt, a następnie zmodyfikować kod programu w taki sposób, aby zapobiec zaistniałej sytuacji.

Litertura

- Grębosz J.,Symfonia C++ standard, Edition 2000, 2008
- Grębosz J., Pasja C++, Edition 2000, 200
- Stroustrup B., Język C++, Wydawnictwa Naukowo-Techniczne 1994, 2000 i 2002
- Eckel B., Thinking in C++. Edycja polska, Helion , 2002