Programowanie II
prowadzący: Adam Dudek

Lista nr 9 (termin 13.05.2021)

Funkcje wirtualne

Funkcje wirtualne sprawiają, że program jest obiektowo zorientowany – inaczej mówiąc orientuje się względem obiektów.

Przykład najprostszy:

```
class instrument
{
 public:
 char nazwa[30];
 void virtual graj()
 {
  cout<<"Intrument wydaje dzwiek .... \n";
 };
};

class trabka:public instrument
{
 public:
  void virtual graj()
  {
   cout<<"Teraz gra trabka .... \n";
  };
};

class fortepian:public instrument
{
  public:
  void virtual graj()
  {
   cout<<"Teraz gra fortepian ...\n";
  };
};

int main(int argc, char* argv[])
{
   instrument jakis_instrument;
   trabka jakas_trabka;
   fortepian jakis_fortepian;
   jakis_instrument.graj();
   jakas_trabka.graj();
   jakis_fortepian.graj();
   Return 0;
}
```

A teraz dla odmiany inna wersja funkcji głównej:

```
   instrument *wsk_na_instrument;
   instrument jakis_instrument;
   trabka jakas_trabka;
   fortepian jakis_fortepian;

   wsk_na_instrument = &jakis_instrument;
```

```
wsk_na_instrument->graj();

wsk_na_instrument = &jakas_trabka;
wsk_na_instrument->graj();

wsk_na_instrument = &jakis_fortepian;
wsk_na_instrument->graj();
```

W praktyce wirtualny oznacza tyle co możliwy, czy też mogący zaistnieć, gdyż funkcja oznaczona jako wirtualna może, ale nie musi być zrealizowana w klasach pochodnych jeszcze raz.

Virtual mówi, że od tej pory po wszystkie dalsze pokolenia kompilator ma użyć swojej „inteligencji", gdy chodzi o wywołania tejże funkcji przez **wskaźnik**.

Polimorfizm
W zależności od wywołania funkcji wirtualnej, wywoływany jest w praktyce inny kod. Zatem np. fragment kodu:

referencja.graj();
wywoływany jest praktycznie w postaci:
referencja.instrument::graj();
lub
referencja.trąbka::graj();
lub
referencja.fortepian::graj();

Mówimy tutaj o wielości formy programu, czyli **polimorfizmie**. Fragment kodu zmienia się tak, że raz odpowiada wywołaniu w wersji pierwszej raz w innej.

Zadanie 1 (30 pkt)
Zaproponuj rozwiązanie, które umożliwiło by przechowywanie obiektów różnych klas (w grze monopoly) w jednej strukturze danych (np. w wektorze). Oczywiście poza listą należałoby zaimplementować możliwość wypełnienia istotnymi danymi wszystkich pól w grze, możliwość ustalenia właściciela danego pola, postawienia na polu domku czy też hotelu, pobrania opłaty (na różnych typach pól) – w praktyce – przeprowadzenia rozgrywki.