1 Wprowadzenie

Polimorfizm to, obok dziedziczenia, najważniejszy mechanizm programowania obiektowego. Pozwala on na tworzenie wielu obiektów posiadających tę samą funkcjonalność ale różniącą się sposobem działania. Przykładem mogą być figury geometryczne, które dostarczają metody obliczającej pole powierzchni. Możemy obliczyć pole powierzchni każdej figury, ale w zależności od rodzaju figury obliczenie go będzie wyglądało inaczej.

```
#include <iostream>

using namespace std;

class Figura{
public:
  float pole(){
    return 0;
  }

};

class Prostokat : public Figura{
private:
  float a, b;
public:
  Prostokat(float a, float b){
    this->a = a;
    this->b = b;
  }

  float pole(){
    return a*b;
  }
};


class Kwadrat : public Figura{
private:
  float a;
public:
  Kwadrat(float a){
    this->a = a;
  }
  float pole(){
    return a*a;
  }
};

class Kolo : public Figura{
private:
  float r;
public:
  Kolo(float r){
```

```
    this->r = r;
  }

  float pole(){
    return 3.14*r*r;
  }
};

int main(int argc, char* argv[]){
Kolo kolo(3);
Prostokat protokat(2,4);
Kwadrat kwadrat(2);

cout << "Pole kola = " << kolo.pole() << endl;
cout << "Pole prostokata = " << protokat.pole() << endl;
cout << "Pole kwadratu = " << kwadrat.pole() << endl;

}
```

Rezultatem programu będzie:

```
Pole kola = 28.26
Pole prostokata = 8
Pole kwadratu = 4
```

Natomiast w przypadku uruchomienia kodu:

```
Figura figury[3] = {Kolo(1), Kwadrat(2), Prostokat(2,3)};
for (int i = 0; i < 3; i++)
  cout << "Pole figury " << i << " = " << figury[i].pole() << endl;
```

Rezultatem programu będzie:

```
Pole figury 0 = 0
Pole figury 1 = 0
Pole figury 2 = 0
```

Ponieważ wykorzystana została tablica obiektów klasy Figura (klasy bazowej) wszystkie figury traktowane są jako klasa bazowa. W związku z tym wykonywana jest metoda klasy bazowej, która zwraca za każdym razem wartość zero a nie odpowiednia metoda z klasy pochodnej. Dzieje się tak dlatego, że na etapie kompilacji kompilator ustawia na stałe adres wykonywanej metody na podstawie typu jakiego używamy (tzw. wczesne wiązanie metod).

Jeśli chcemy aby wykonywana została wykonywana metoda właściwego obiektu musimy odroczyć ustawienie adresu do metody do czasu wykonywania programu (tzw. późne wiązanie). W języku C++ wykonujemy to zadanie za pomocą słowa kluczowego virtual.

```
class Figura{
public:
  virtual float pole(){
    return 0;
  }

};

class Prostokat : public Figura{
private:
  float a, b;
public:
  Prostokat(float a, float b){
    this->a = a;
    this->b = b;
  }

  float pole(){
    return a*b;
  }
};


class Kwadrat : public Figura{
private:
  float a;
public:
  Kwadrat(float a){
    this->a = a;
  }
  float pole(){
    return a*a;
  }
};

class Kolo : public Figura{
private:
  float r;
public:
  Kolo(float r){
    this->r = r;
  }

  float pole(){
    return 3.14*r*r;
  }
};
```

W tym przypadku deklarujemy, że metoda pole() jest metodą wirtualną, więc adres metody do wykonania będzie obliczany nie na etapie kompilacji ale na etapie działania programu (dynamicznie). Dzięki temu będzie mogła zostać wykonany za każdym razem odpowiedni kod.

Wykonanie następującego kodu (UWAGA!: Należy zwrócić uwagę, że tym razem użyto wskaźników na obiekty a nie samych obiektów!):

```
Figura* figury[3] = { new Kolo(1), new Kwadrat(2), new Prostokat(2,3)};
for (int i = 0; i < 3; i++)
  cout << "Pole figury " << i << " = " << figury[i]->pole() << endl;
```

spowoduje w rezultacie:

```
Pole figury 0 = 3.14
Pole figury 1 = 4
Pole figury 2 = 6
```

2 Zadania do wykonania

1. Sprawdzić możliwość wykorzystania metod wirtualnych na tablicy obiektów

2. Sprawdzić możliwość wykorzystania metod wirtualnych na tablicy wskaźników na obiekty

3. Sprawdzić możliwość wykorzystania metod wirtualnych na kolekcji vector obiektów

4. Sprawdzić możliwość wykorzystania metod wirtualnych na kolekcji vector wskaźników na obiekty

5. Przygotować zestaw klas reprezentujących gatunki i konkretne zwierzęta. Każde zwierze powinno poruszać się, dawać głos na swój własny sposób

6. Stworzyć system do zarządzania zwierzętami z poprzedniego zadania. System powinien mieć możliwość dodawania nowych zwierząt, usuwania, edycji oraz poruszania i dawania głosu przez wszystkie zwierzęta naraz