

C++/Czym jest obiekt

Aby odpowiedzieć na pytanie zadane w temacie, zadajmy sobie inne:

Co nazywamy obiektem w świecie rzeczywistym?

Otóż wszystko może być obiektem! Drzewa, zwierzęta, miasta, auta, ludzie...

W programowaniu również obiektem może być dowolny twór, o jakim pomyślimy. Tworząc "świat programu" można stworzyć obiekt, którego użycie będzie bardziej "namacalne" od szeregu parametrów, porzucanych w różnych zmiennych. To różni *programowanie strukturalne* od *programowania obiektowego*.

Projekt i twór – klasa i obiekt

Zanim stworzymy jakiś obiekt, trzeba ustalić czym ten obiekt będzie. W zależności od tego, czy chcemy stworzyć wirtualny samochód, czy samolot, należy określić dwie rzeczy:

- jakie **właściwości** będzie miał ten obiekt,
- jakie będzie miał **metody** działania.

W związku z tym, przed stworzeniem jakiegokolwiek obiektu należy przedstawić kompilatorowi jego **projekt**(wzorzec), czyli określić jego **klasę**.



Klasa to byt programistyczny określający jakie *właściwości* i *metody* będą miały obiekty, które zostaną utworzone na jej podstawie.

Jednak sam projekt nie sprawi jeszcze, że dostaniemy obiekty (to tak jakby po narysowaniu projektu domu chcieć zamieszkać na kartce papieru :-)). Trzeba jeszcze obiekt **utworzyć**, co oznacza po prostu **deklarację** obiektu na podstawie pewnej *klasy*:

```
NazwaKlasy MojObiekt;
```

Wygląda to jak deklaracja zwykłej zmiennej i tak jest w istocie – w C++ tworząc klasę definiuje się **nowy typ danych**. Podobnie jak w przypadku zmiennych, można utworzyć **wiele** obiektów danej klasy.

Definicja klasy

Ogólny szablon definiowania klas w C++ wygląda następująco:

```
class NaszaNazwaKlasy {  
    ... // pola i metody składowe klasy  
};
```

Po słowie kluczowym **class** następuje nazwa naszej klasy (prawidłą jej nazywania są takie same jak dla zmiennych). W nawiasach klamrowych umieszcza się definicje składowych klasy: **pól** i **metod** określając dla nich specyfikatory dostępu.

Należy pamiętać o średniku za klamerką zamykającą definicję klasy.

Oto przykładowa definicja klasy:

```
class NazwaKlasy {  
    public: //pola i metody są publicznie dostępne  
  
    //definiowanie pól  
    int poleInt;  
    float poleFloat;
```

```
//deklarowanie metod
int Metoda1();
void Metoda2();

}; //pamiętaj o średniku!
```

Użycie klasy

Sama definicja klasy nie wystarczy, aby uzyskać dostęp do jej składowych. Należy stworzyć obiekt. Można przyjąć, że obiekt to zmienna typu klasowego. Deklaracja obiektu:

```
NazwaKlasy Obiekt;
```

Dostęp do pól i metod uzyskuje się operatorem (.):

```
Obiekt.poleInt = 0; //przypisanie wartości polom
Obiekt.poleFloat = 9.04;
Obiekt.Metoda1(); //wywołanie metody obiektu
```

W przypadku deklaracji wskaźnika do obiektu:

```
NazwaKlasy *ObiektWsk = new NazwaKlasy;
```

Analogicznie jak w przypadku wskaźników na struktury operatorem dostępu do pola/metody klasy poprzez wskaźnik do obiektu staje się ->:

```
ObiektWsk->poleInt = 0; //przypisanie wartości polom
ObiektWsk->poleFloat = 9.04;
ObiektWsk->Metoda1(); //wywołanie metody obiektu
```

Należy pamiętać o zniszczeniu obiektu przed zakończeniem działania programu (lub kiedy nie jest nam już potrzebny):

```
delete ObiektWsk;
```

Przykład

Stwórzmy klasę kostki do gry:

```
class Kostka{
public:
    unsigned int wartosc;
    unsigned int maks;
    void Losuj();
};
```

Po definicji klasy, zdefiniujemy jeszcze metodę *Losuj()* zadeklarowaną w tej klasie:

```
void Kostka::Losuj()
{
    wartosc = rand()%maks + 1;
}
```

Warto zwrócić uwagę w jaki sposób się to robi. Nazwą metody dowolnej klasy jest *NazwaKlasy::NazwaMetody*. Poza tym aby uzyskać dostęp do pól klasy, w której istnieje dana metoda nie stosuje się operatora wyłuskania.

Po tym można napisać resztę programu:

```
int main()
{
    Kostka kostkaSzescienna; //utworzenie obiektu
    kostkaSzescienna.maks = 6; //określenie maksymalnej ilości oczek
    kostkaSzescienna.Losuj(); //losowanie
    cout << "Wylosowano:" << kostkaSzescienna.wartosc << endl; //wypisanie wyniku
    return 0;
}
```

Autorekursja

Wskaźnik **this** umożliwia jawne odwołanie się zarówno do atrybutów, jak i metod klasy. Poniższy program wymusza użycie wskaźnika *this*, gdyż nazwa pola jest taka sama jak nazwa argumentu metody *wczytaj*:

```
#include <iostream>

class KlasaThis {
    int liczba;
public:
    void wczytaj(int liczba) {this->liczba=liczba;}
    void wypisz() {cout << liczba <<endl;}
};

int main()
{
    KlasaThis ObiektThis;
    ObiektThis.wczytaj(11);
    ObiektThis.wypisz();

    return 0;
}
```

Kontrola dostępu

Istnieją trzy specyfikatory dostępu do składowych klasy:

- **private** (prywatny) - dostępne tylko z wnętrza danej klasy i klas/funkcji zaprzyjaźnionych.
- **protected** (chroniony) - dostępne z wnętrza danej klasy, klas/funkcji zaprzyjaźnionych i klas pochodnych.
- **public** (publiczny) - dostępne dla każdego.

Jeśli sekwencja deklaracji składowych klasy nie jest poprzedzona żadnym z powyższych specyfikatorów, to domyślnym specyfikatorem (dla kompilatora) będzie **private**.

Dzięki specyfikatorom dostępu inni programiści mają ułatwione korzystanie z utworzonej przez nas klasy, gdyż metody i pola, których nie powinni modyfikować, bo mogłoby to spowodować niepoprawne działanie obiektu, są oznaczone jako **private** lub **protected** i nie mogą z nich korzystać. Funkcje, które zapewniają pełną funkcjonalność klasy oznaczone są jako **public** i tylko do nich ma dostęp użytkownik klasy (do **protected** również, ale z ograniczeniami). Oto zmodyfikowany przykład z kostką, który zobrazuje cele kontroli dostępu:

```
class Kostka{
public :
    void Losuj();
    void Wypisz();
    int DajWartosc();
    void ZmienIloscScian(unsigned argMax);
protected:
    unsigned wartosc;
    unsigned max;
};

int Kostka::DajWartosc()
{
    return this->wartosc;
}

void Kostka::ZmienIloscScian(unsigned argMax)
{
    if(argMax > 20)
        max = 20;
    else
        max = argMax;
}
```

Zmodyfikowana klasa zezwala tylko na kostki maksymalnie dwudziestościenne. Ręczne modyfikacje zmiennej max są zabronione, można tego dokonać jedynie poprzez funkcję ZmienIloscScian, która zapobiega przydzieleniu większej ilości ścianek niż 20. Prywatny jest też atrybut wartość. Przecież nie chcemy aby była ona ustawiona inaczej niż przez losowanie! Dlatego możemy udostępnić jej wartość do odczytu poprzez metodę DajWartosc(), ale modyfikowana może być tylko na skutek działania metody Losuj().

Ćwiczenia

1. Napisz klasę reprezentującą człowieka. Musi on być opisany przy pomocy: imienia, nazwiska, płci, wieku, partnerki/partnera(jako wskaźnik).
2. Rozwiń klasę napisaną w 1. ćwiczeniu dodając ograniczenie, między partnerami nie może być większa niż 25% wieku starszej z nich.
- 3*. Jeśli zaznajomiłeś się z wektorami, dodaj kolejny parametr opisujący ludzi - zainteresowania, dodaj odpowiednią funkcję do dodawania nowych zainteresowań do listy oraz funkcję porównującą zainteresowania obu ludzi i zwracającą procent identycznych zainteresowań.

Źródła i autorzy artykułu

C++/Czym jest obiekt Źródło: <http://pl.wikibooks.org/w/index.php?oldid=171597> Autorzy: Albmont, Derbeth, Felix, Fidytek, Gang65, Kj, Lethern, Mythov, PGK, Piotr, 26 anonimowych edycji

Źródła, licencje i autorzy grafik

Grafika:Plume ombre.png Źródło: http://pl.wikibooks.org/w/index.php?title=Plik:Plume_ombre.png Licencja: GNU Free Documentation License Autorzy: Darkdadaah, Javierme, Rilegator, Rocket000

Licencja

Creative Commons Attribution-Share Alike 3.0 Unported
[//creativecommons.org/licenses/by-sa/3.0/](http://creativecommons.org/licenses/by-sa/3.0/)
