1 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.

2 Definicja klasy

W definicji klasy w C++ wyodrębnia się cztery następujące elementy:

- Zbiór, który może być pusty, danych składowych - zwanych **polami**. Pola są wewnętrzną reprezentacją klasy; każde z nich może mieć dowolny typ.
- Zbiór, który może być pusty, funkcji składowych - zwanych **metodami**. Metody określają dozwolone operacje dotyczące obiektów danej klasy, a więc sposób, w jakim używa się tych obiektów; mówi się też, że określają „metodę" komunikowania się obiektów - między sobą oraz z resztą programu.
- **Poziomy dostępu do składowych** - Specyfikatory *private, protected* oarz *public* określają, które składowe klasy są prywatne, które chronione, a które publiczne.

- **Nazwa klasy**, która służy jako specyfikator typu zdefiniowanego przez użytkownika. Nazwa klasy może pojawić się we wszystkich tych miejscach programu, w których może pojawić się specyfikator typu zdefiniowanego pierwotnie.

Ogólny szablon definiowania klas w C++ wygląda następująco:

```
class NaszaNazwaKlasy {
  // pola i metody skladowe klasy
     };
```

Po słowie kluczowym class następuje nazwa naszej klasy (prawidła 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. Przykładowa definicja klasy:

```
class NazwaKlasy {
       public: //pola i metody sa publicznie dostepne

    //definiowanie pol
         int poleInt;
         float poleFloat;

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

    }; //pamietaj o sredniku!
```

2.1 Użycie klasy

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

3 Deklaracja obiektu

```
NazwaKlasy Obiekt;
//Dostep do pol i metod uzyskuje sie operatorem (.):
Obiekt.poleInt = 0;//przypisanie wartosci polom
Obiekt.poleFloat = 9.04;
Obiekt.Metoda1();//wywolanie metody obiektu
```

3.1 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 wartosci polom
ObiektWsk->poleFloat = 9.04;
ObiektWsk->Metoda1(); //wywolanie metody obiektu
```

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

```
delete ObiektWsk;
```

4 Przykład

Stwórzmy klasę kostki do gry:

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

Po definicji klasy, zdefiniujmy 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; //okreslenie maksymalnej ilosci oczek
 kostkaSzescienna.Losuj(); //losowanie
 cout << "Wylosowano:" << kostkaSzescienna.wartosc << endl;//wypisanie wyniku
 return 0;
}
```

4.1 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;
}
```

4.2 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). 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().

5 Zadania do wykonania

1. Przygotuj klasę zawierającą pola i metody dla: Samochodu, Telefonu i Samolotu.
2. Przygotuj klasę zawierającą prostą implementację stosu.
3. Przygotuj klasą zawierającą pola i metody dla Osoby.
4. Za pomocą klasy Vector stwórz kolekcję Osób.
5. Zapoznaj się z metodą przeszukiwania kolecji za pomocą iteratorów.
6. Przygotuj metody pozwalające na wyszukiwanie Osób po imionach, nazwiskach, adresach itd. z kolekcji Osób.