Programowanie komputerowe

Zajęcia 7

Klasy

- Klasy to typy danych, które pozwalają na zgromadzenie w jednej zmiennej (obiekcie) zarówno danych jak i operacji związanych z tymi danymi.
- Obiekt danej klasy może zawierać zmienne (zwane **polami**) i funkcje (zwane **metodami**).
- Można ustalić które składniki (pola i metody) obiektów są dostępne dla pozostałej części programu (składniki publiczne), a które są niedostępne (składniki prywatne).
- Można tworzyć własne klasy na podstawie już istniejących – ten mechanizm nazywa się dziedziczeniem.

Definicja klasy

```
class NazwaKlasy {
public: // poniższe składniki będą publiczne
    deklaracja_zmiennej_lub_funkcji
    …
    deklaracja_zmiennej_lub_funkcji
private: // a teraz składniki prywatne
    deklaracja_zmiennej_lub_funkcji
    …
    deklaracja_zmiennej_lub_funkcji
}; // tu jest średnik!!!
```

Deklaracja klasy – przykład

Poniższa klasa umożliwia przechowywanie punktów na płaszczyźnie:

```
class Punkt {
public:
    double x; // współrzędna x
    double y; // współrzędna y
};
```

Uwaga: jeśli wszystkie składniki są prywatne możemy użyć struct zamiast class:

```
struct Punkt { double x; double y; }
```

Przykład użycia klasy Punkt

```
int main() {
    Punkt p;    // tworzymy nowy punkt
    p.x=2.5;    // ustawiamy współrzędne
    p.y=1.0;
    cout << "Punkt p to (" << p.x << "," << p.y << ")";
    p.x=p.x+1;  // przesuwamy w prawo
    cout << "A teraz p to (" << p.x << "," << p.y << ")";
}
```

Jak widać powyżej, do składnika `x` obiektu `p` odwołujemy się pisząc `p.x`.

Poprawiamy klasę Punkt

Dodamy metodę do klasy Punkt, która umożliwia drukowanie współrzędnych punktu na ekranie:

```
class Punkt {
public:
    double x; // współrzędna x
    double y; // współrzędna y
    void drukuj(); // drukuje współrzędne na
ekranie-deklaracja
};
void Punkt::drukuj() {  // definicja
    cout << "(" << x << "," << y << ")";
}
```

Poprawiamy klasę Punkt (2)

Można też połączyć definicję z deklaracją:

```
class Punkt {
public:
    double x; // współrzędna x
    double y; // współrzędna y
    void drukuj() { // drukuje współrzędne na ekranie
        cout << "(" << x << "," << y << ")";
    }
};
```

Tak robimy jeśli funkcja jest krótka.

Poprawiamy klasę Punkt (3)

Teraz punkty "potrafią" się drukować. Można poprzedni przykład zapisać tak:

```
int main() {
    Punkt p;    // tworzymy nowy punkt
    p.x=2.5;    // ustawiamy współrzędne
    p.y=1.0;
    cout << "Punkt p to "; p.drukuj();
    p.x=p.x+1;  // przesuwamy w prawo
    cout << "A teraz p to "; p.drukuj();
}
```

Później nauczymy się drukować obiekty przy pomocy cout.

Konstruktory

- Konstruktor to funkcja, która automatycznie wywołuje się w momencie, gdy tworzony jest nowy obiekt.
- Konstruktory służą do ustawienia początkowej wartości obiektów.
- W klasie może być zdefiniowanych wiele konstruktorów, muszą się różnić listą parametrów. Gdy tworzymy obiekt możemy wybrać, którego z nich użyć.
- Deklaracja konstruktora wygląda jak definicja funkcji, ale:
  - nazwa konstruktora jest taka sama jak nazwa funkcji
  - konstruktor nie zwraca żadnej wartości

Konstruktory klasy Punkt

```
class Punkt {
public:
    Punkt() { x=0; y=0; }   // konstruktor 1
    Punkt(double _x, double _y) { // konstruktor 2
        x=_x; y=_y; // ustawiamy początkowe współrzędne
    }
    double x; // współrzędna x
    double y; // współrzędna y
    void drukuj() { // drukuje współrzędne na ekranie
        cout << "(" << x << "," << y << ")";
    }
};
```

Konstruktory – przykład

```
int main() {
    Punkt p(2.5, 1.0);   // tworzymy nowy punkt – konstruktor 1
    Punkt q;             // tworzymy nowy punkt – konstruktor 2
    cout << "Punkt p to ";
    p.drukuj();
    cout << "Punkt q to ";
    q.drukuj();
}
```

Przykład – metoda licząca odległość

Do klasy Punkt dodajemy:

```
double odleglosc(Punkt& p) {
    return sqrt((x-p.x)*(x-p.x)+(y-p.y)*(y-p.y));
}
```

(wymaga biblioteki cmath). Teraz wyrażenie

```
p.odleglosc(q)
```

zwróci odległość od punktu p do punktu q.
Uwaga: funkcja odleglosc ma dwa parametry – jeden wpisany w nawiasy, a drugi niejawny – obiekt wywołujący funkcję.

Ćwiczenia

1. Dodać do klasy Punkt metodę

   ```
   void wczytaj()
   ```

   która umożliwia użytkownikowi wpisanie współrzędnych punktu.
2. Stworzyć klasę Czas, której obiekty będą reprezentować czas w ciągu dnia z dokładnością do minuty. Należy umożliwić wypisywanie godziny w formatach 12 i 24-godzinnym oraz dodawanie do niej podanej ilości minut.
3. Stworzyć klasę Set, która reprezentuje wynik meczu pomiędzy dwoma drużynami. Set toczy się do zdobycia określonej liczby punktów i osiągnięcia dwupunktowej przewagi nad przeciwnikiem. Dostępne metody: zdobycie punktu, drukowanie wyniku, określenie zwycięzcy.