Programowanie komputerowe

Zajęcia 10

Konstruktor kopiujący

Jeśli istnieje potrzeba stworzenia kopii obiektu klasy T używany jest konstruktor kopiujący:

```
T(const T& obiekt)
```

Słowo `const` oznacza, że konstruktor ten nie może zmienić oryginału, a referencja jest niezbędna aby nie tworzyć kopii parametru.

Jeśli konstruktor kopiujący nie jest zdefiniowany, kopia obiektu powstaje przez skopiowanie wartości poszczególnych pól. Nie zawsze jest to właściwe, np. gdy obiekt zawiera tablice.

Przeciążenie operatora przypisania

Jeśli `x` i `y` są obiektami klasy `T`, przypisanie

```
x=y;
```

powoduje skopiowanie wszystkich pól `y` na pola `x`. Jeśli chcemy, żeby ta operacja była wykonywana w inny sposób, musimy przeciążyć operator przypisania, np. tak:

```
T& operator=(const T& obiekt);
```

Zwyczajowo przypisanie zwraca obiekt na który przypisujemy wartość. Dostęp do obiektu wykonującego metodę zapewnia wskaźnik `this`, więc ten obiekt to `*this`.

Listy inicjujące

Istnieje możliwość bezpośredniego podania wartości zmiennych w klasie podczas tworzenia obiektu, np. tak:

```
class A {
public:
    A(): liczba(7), napis("SIEDEM") {}
    int liczba;
    string napis;
};
```

Wartości podane w nawiasach są użyte do ustawienia początkowej wartości, tak więc liczba przyjmie wartość `7`, a napis zostanie stworzony przy pomocy konstruktora z parametrem `"SIEDEM"`.

Destruktory

W klasie można zdefiniować destruktor – jest to funkcja automatycznie wywoływana automatycznie bezpośrednio przez zniszczeniem obiektu. Zazwyczaj usuwa ona tablice stworzone przez obiekt. Destruktor ma nazwę taką jak klasa poprzedzoną tyldą. Przykład:

```
class Lista {
public:
    Lista() { t=new int[10]; }
    ~Lista() { delete [] t; }
    int* t;
}
```

Tablica automatycznie się usunie kiedy obiekt przestanie istnieć.

Ćwiczenia

1. Stworzyć klasę, której obiekty reprezentują napisy.
2. Stworzyć klasę, której obiekty reprezentują nieskończone tablice (tj. o indeksach 0,1,2,...).
3. Stworzyć klasę `Macierz`, której obiekty reprezentują macierze 3x3 o współczynnikach typu `double`. Należy udostępnić standardowe operacje na macierzach, takie jak ustawianie współczynników, drukowanie, dodawanie, mnożenie, liczenie wyznacznika, transpozycja. Należy też zadbać o to, aby przypisanie macierzy działało poprawnie (tj. żeby współczynniki tych macierzy były w odrębnych tablicach).

Zadania (2)

4. Stworzyć klasę `Kolejka`, która przechowuje ciąg napisów (`string`).

```
class Kolejka {
public:
    Kolejka(); // pusta kolejka
    void dodaj(string s); // dodaje napis na końcu
    string usun(); // usuwa pierwszy element i go zwraca
    void drukuj(); // drukuje wszystkie elementy
    bool pusta(); // zwraca true jeśli kolejka jest pusta
};
```

Zadania (3)

5. Stworzyć klasę `ZbiorLiczb`, która przechowuje zbiory liczb typu `int`.

```
class ZbiorLiczb {
public:
    ZbiorLiczb(); // pusty zbior
    void dodaj(int a); // dodaje liczbę
    void usun(int a); // usuwa liczbę
    bool nalezy(int a); // true jeśli element należy do zbioru
    void dodaj(const ZbiorLiczb& z); //dodaje elementy zbioru z
    int ile(); // zwraca liczbę elementów
};
```

Zadania (4)

6. Zaimplementować klasę `Histogram`, która zapamiętuje wyniki pomiarów w podziale na zadane przedziały.

```
class Histogram {
public:
    Histogram(double min, double max, int ilePrzedzialow); // Tworzy nowy obiekt, dla wartości z przedziału [min,max] podzielone na ilePrzedzialow równych części.
    int dodaj(double wartosc); // Dodaje nową wartość.
    int ileWPrzedziale(int numer); // Zwraca liczbę wartości w podanym przedziale
    int ilePozaZakresem(); // Zwraca liczbę wartości poza obsługiwanym zakresem.
    void drukuj(); // drukuje histogram
};
```

   Następnie zmodyfikować tą klasę tak, aby do liczby pomiarów w danym przedziale odwoływać się przez [].