C++

Lab_2

Konstruktor kopiujący a konstruktor przenoszący

1. Utwórz nowy projekt i dodaj plik node_coord.h, który zawiera definicję klasy NODE_COORD

```
class NODE_COORD
{
	double *pcoord;  //pcoord[0] - x, pcoord[1] - y
public:
	NODE_COORD() : pcoord(NULL) {}    //default constructor
	NODE_COORD(double x, double y);   //parameterized constructor
	~NODE_COORD();                    //destructor
	void disp();                      //output x, y to screen
private:
	void crash();                     // handles memory allocation error
};
```

Dodaj do projektu implementację metod klasy NODE_COORD w pliku *.cpp:

```
NODE_COORD::NODE_COORD(double x, double y)
{
    try
    {
	pcoord = new double [2];
	pcoord[0] = x;
	pcoord[1] = y;
    }
    catch(bad_alloc)
    {
	crash();
    }
}


void NODE_COORD::crash()
{
    cout << "memory allocation error\n";
    system("pause");
    exit(1);
}


NODE_COORD::~NODE_COORD()
{
    if(pcoord)
    {
	delete [] pcoord;
	pcoord = NULL;
    }
}
```

Funkcja main wykonuje następujące:

```
int main()
{
    NODE_COORD A(2, 3);
    NODE_COORD B = A;

    NODE_COORD C;
    NODE_COORD D = C;

    system("pause");
    return 0;
}
```

Uruchom program, wyjaśnij przyczynę niepowodzenia i popraw kod, aby działał poprawnie.

Dodaj klasę Triangle (pliki triangle.h i triangle.cpp), która zawiera trzy wierzchołki NODE_COORD vert_A, NODE_COORD vert_B, NODE_COORD vert_C i wiersz tekstowy char str[128], który przechowuje nazwę trójkąta (na przykład „trójkąt ABC"). Użyj sparametryzowanego konstruktora do wprowadzania danych. Nie używaj operatora przypisania „=". Utwórz metodę disp() do wyświetlania danych na monitorze, używając metody disp() klasy NODE_COORD dla każdego wierzchołka. Umieść definicję klasy Triangle w pliku triangle.h, a implementację metod klasy w pliku triangle.cpp. Funkcja main() wygląda następująco:

```
int main()
{
    NODE_COORD A(2, 3);
    NODE_COORD B = A;

    NODE_COORD C;
    NODE_COORD D = C;

    NODE_COORD AA(2, 3), BB(3,4), CC(0, 0);
    Triangle tr(AA, BB, CC, "triangle 1");
    tr.disp();

    system("pause");
    return 0;
}
```

2. Dodaj metodę double distance(int First, int Second) w klasie Triangle, która oblicza odległość między wierzchołkami First i Second (długość boku FirstSecond). Zmienna pcoord klasy NODE_COORD musi być prywatna. W tym celu w pliku node_coord.cpp utwórz oddzielną funkcję (nie metodę żadnej klasy) double distance (NODE_COORD A, NODE_COORD B), i zaprzyjaźnij ją z klasą NODE_COORD (friend double distance (NODE_COORD A, NODE_COORD B);). Metoda distance klasy Triangle musi wywoływać metodę double NODE_COORD :: distance(NODE_COORD A, NODE_COORD B). Funkcja main() wygląda następująco

```
int main()
{
    NODE_COORD A(2, 3);
    NODE_COORD B = A;
```

```
	NODE_COORD C;
	NODE_COORD D = C;

	NODE_COORD AA(2, 3), BB(3, 4), CC(0, 0);
	Triangle tr(AA, BB, CC, "trojkat 1");
	tr.disp();
	cout << "distance between first and second nodes: " << tr.distance(1, 0) <<
endl;
	system("pause");
	return 0;
}
```

3. Utwórz funkcję o nazwie void fun (Triangle trr), która pobiera obiekt typu Triangle i wywołuje metodę disp() w celu wyświetlenia jego na monitorze. Przeciąż tą funkcję jako void fun (Triangle * trr), używając tylko wskaźnika do obiektu Triangle. Funkcja main wygląda teraz tak

```
int main()
{
    NODE_COORD A(2, 3);
    NODE_COORD B = A;

    NODE_COORD C;
    NODE_COORD D = C;

    NODE_COORD AA(2, 3), BB(3,4), CC(0, 0);
    Triangle tr(AA, BB, CC, "trojkat 1");
    tr.disp();

    my_fun(tr);
    my_fun(&tr);

    system("pause");
    return 0;
}
```

5. Utwórz funkcję o nazwie `Triangle fun(NODE_COORD& A, NODE_COORD& B, NODE_COORD& C)`, która pobiera referencje do wierzchołków A, B, C, tworzy lokalny obiekt Triangle i zwraca jego. Funkcja main wygląda teraz tak:

```
#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>

int main()
{
	{
		NODE_COORD A(2, 3);
		NODE_COORD B = A;

		NODE_COORD C;
		NODE_COORD D = C;

		NODE_COORD AA(2, 3), BB(3, 4), CC(0, 0);
		Triangle tr(AA, BB, CC, "trojkat 1");
		tr.disp();
```

```
            cout << "distance between first and second nodes: " << tr.distance(1,
0) << endl;

            Triangle ttrr(tr);
            ttrr.disp();

            cout << "distance between first and second nodes: " << ttrr.distance(0,
1) << endl;

            fun(ttrr);
            fun(&ttrr);

            Triangle ttrr1(fun(AA, BB, CC));
            ttrr1.disp();
      }

      if (_CrtDumpMemoryLeaks())
            cerr << "LEAK MEMORY !!!!!!!!!!!!!!!!!!!!\n";

      return 0;

}
```

Użyj debugera, aby wyjaśnić wywołanie konstruktorów i destruktorów klas NOD_COORD i Triangle:

- przy wywoływaniu pierwszej, drugiej i trzeciej wersji funkcji fun.
- kiedy wywołuje się konstruktor kopiujący a kiedy – konstruktor przenoszący.