C++

Lab_2

Copy constructor

1. Create a new project and add the file node_coord.h, which has a definition of the NODE_COORD class:

```
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
};
```

The implementation of methods of NODE_COORD class put to file *.cpp (add to project):

```
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;
    }
}
```

Function main makes the following:

```
int _tmain(int argc, _TCHAR* argv[])
{
    NODE_COORD A(2, 3);
    NODE_COORD B = A;

    NODE_COORD C;
    NODE_COORD D = C;

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

Run the program, explain the cause of the failure, and correct the code so that it runs properly.

2. Add *Triangle* class (files triangle.h and triangle.cpp), which contains three vertices NODE_COORD vert_A, NODE_COORD vert_B, NODE_COORD vert_C and string str[128], which stores the name of triangle (for instance, "triangle ABC"). Use the parameterized constructor for data input. Do not use the assignment operator "=". Create the method disp() for displaying data on the monitor, using the method disp() of class NODE_COORD for each vertex. Place a definition of the *Triangle* class in file triangle.h as well as the implementation of the class methods in the file triangle.cpp. The function main() seems as following:

```
int _tmain(int argc, _TCHAR* argv[])
{
    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;
}
```

3. Add the method *double distance*(*int First*, *int Second*) in class *Triangle*, which calculates the distance between vertices First and Second (the length of a side *FirstSecond*). Variable *pcoord* of the NODE_COORD class must be *private*. For this, in the node_coord.cpp file create alone function (not a method of any class) *double distance* (NODE_COORD A, NODE_COORD B) and friend it to the NODE_COORD class (*friend double distance* (NODE_COORD A, NODE_COORD B);). Method *distance* of the *Triangle* class must call method *double* NODE_COORD :: *distance*(NODE_COORD A, NODE_COORD B). The function main() seems as following:

```
int _tmain(int argc, _TCHAR* argv[])
{
    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;
}
```

4. Create a function called *void fun* (*Triangle* trr) which takes an object of type *Triangle* and calls disp() method for display on the monitor. Override this function *void fun* (*Triangle* * trr) and do the same, just using a pointer to the Triangle object. The main function looks like this now:

```
int _tmain(int argc, _TCHAR* argv[])
{
    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;
}
```

Use debugger to explain the difference when calling the first and second functions.