Tutorial n°4 - Classes, objects, and pointers

We are going to develop simple (and naïve) structures to handle 2D closed polygons. As we are going to see through this example, we do not necessarily need to handle arrays when allocating the memory. Actually, the corresponding data structure created will be a double chained list.

1 2D Point

1. Declare and implement a class *Point2d* to represent a $2D$ point which has two private members x and y coded by floats.
2. Implement a member function *display*(...) to display the values of x and y. Then, you can also try to overload the operator « for this class such as you could type statements such as `cout « myPoint2D «endl;`
3. Declare and implement a function *set*(...) to modify the values of the members x and y for a $2D$ point (by pointer and by reference). You can also implement a function *askvalue*(...) to ask the values to the user.
4. Declare and initialize a "dummy" *Point2d* within the main function to test your results.

2 Polygon

To represent a polygon, we need to handle several $2D$ points. Additionally, we need to know the order of the points within the polygon. To do so, update the class *Point2d* by adding two private members *prev* and *next*, that are pointers to the predecessor and the successors of a point within a polygon.

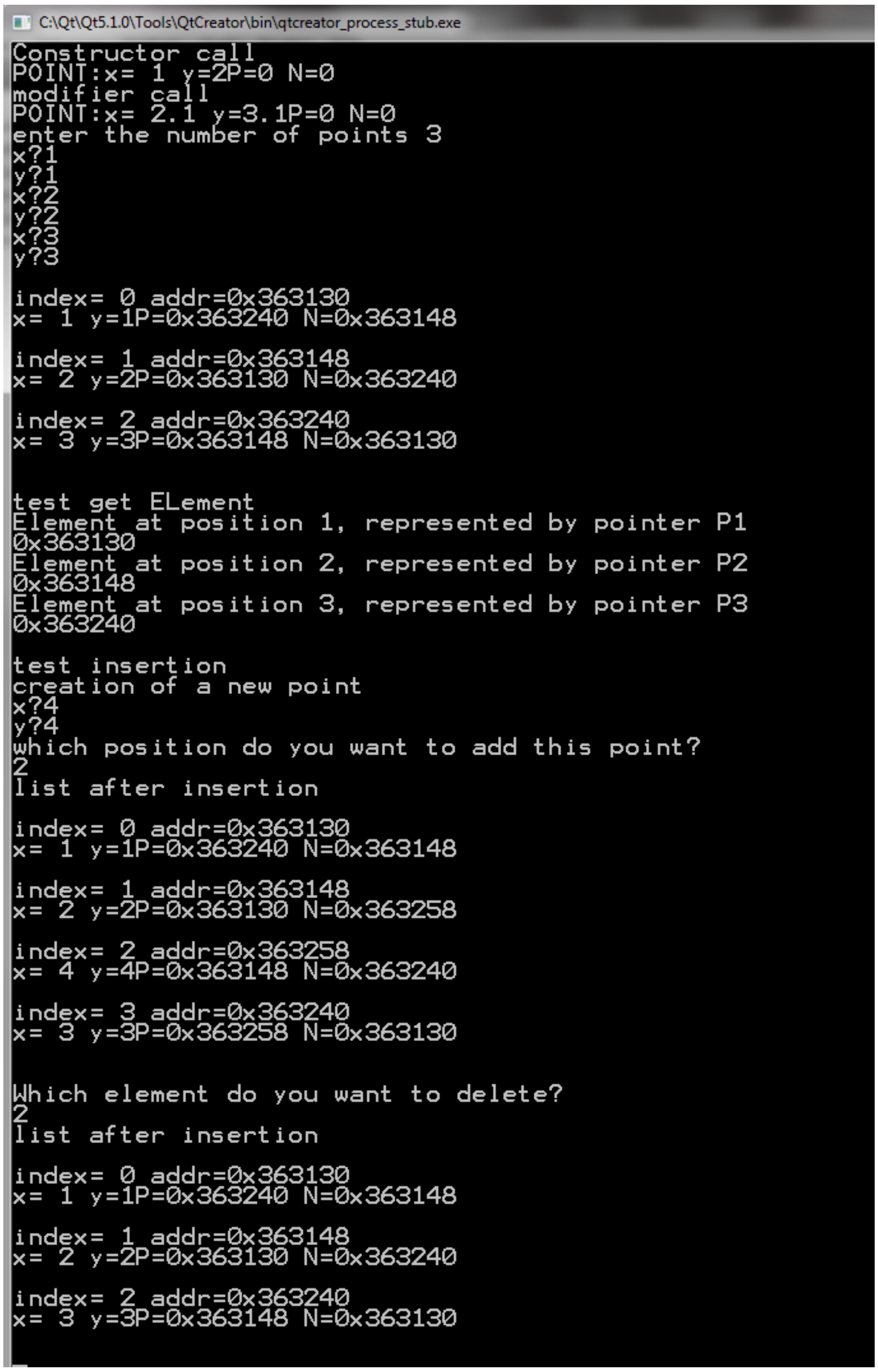
1. Declare and implement a class *Polygon* that contains a pointer to an initial *Point2d* called *start*.
2. Declare and implement a function *BuildPolygon*(...) to which is provided the number of points of the polygon. The function then asks for the coordinates of each point (within the order of the polygon), and then creates and inserts the corresponding $2D$ within the polygon.
3. Declare and implement a function that displays the elements of a polygon. This function should also display the previous and next points within the polygon. Same question as before, here you can try to overload the operator « for the class *Polygon*

3 Insertion and deletion of elements

Declare and implement the following functions:

1. *begin*() that returns a pointer to the first element;
2. *size*() that returns the number of points in the polygon.
3. *get_item*(...) that returns a pointer to a $2D$ Point at position in a given polygon.
4. *insert_at*(...) that inserts an element at a given position in the list.
5. *delete_at*(...), that deletes (if possible) an element at position I.
6. Overload the operator [] for such class.

You should obtain these kinds of results:



```
C:\Qt\Qt5.1.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
Constructor call
POINT:x= 1 y=2P=0 N=0
modifier call
POINT:x= 2.1 y=3.1P=0 N=0
enter the number of points 3
x?1
y?1
x?2
y?2
x?3
y?3

index= 0 addr=0x363130
x= 1 y=1P=0x363240 N=0x363148

index= 1 addr=0x363148
x= 2 y=2P=0x363130 N=0x363240

index= 2 addr=0x363240
x= 3 y=3P=0x363148 N=0x363130


test get ELement
Element at position 1, represented by pointer P1
0x363130
Element at position 2, represented by pointer P2
0x363148
Element at position 3, represented by pointer P3
0x363240

test insertion
creation of a new point
x?4
y?4
which position do you want to add this point?
2
list after insertion

index= 0 addr=0x363130
x= 1 y=1P=0x363240 N=0x363148

index= 1 addr=0x363148
x= 2 y=2P=0x363130 N=0x363258

index= 2 addr=0x363258
x= 4 y=4P=0x363148 N=0x363240

index= 3 addr=0x363240
x= 3 y=3P=0x363258 N=0x363130


Which element do you want to delete?
2
list after insertion

index= 0 addr=0x363130
x= 1 y=1P=0x363240 N=0x363148

index= 1 addr=0x363148
x= 2 y=2P=0x363130 N=0x363240

index= 2 addr=0x363240
x= 3 y=3P=0x363148 N=0x363130
```