Software Engineering
Lab 4 Report

Emre Ozan Alkan
{emreozanalkan@gmail.com}
MSCV-5

1 November 2013

1 2D Point

1.1 Declare and implement Point2d class

Listing 1: Point2d.h

```
1  #ifndef POINT2D_H
2  #define POINT2D_H
3
4  #include <iostream>
5
6  // Representing 2D point
7  // 2 points represented by float values x and y
8  // Has pointers to previous and next elements
9  // to be used in chain list data structure
10 class Point2d
11 {
12 private:
13     // Represents the value of x coordinate of 2D point
14     float _x;
15     // Represents the value of y coordinate of 2D point
16     float _y;
17
18     // Keeps track of the previous element in chain list
19     Point2d* _prev;
20     // Keeps track of the next element in chain list
21     Point2d* _next;
22 public:
23     // Default Constructor
24     Point2d();
25     // Overloaded Constructor for initializing values _x and _y
26     Point2d(float, float);
27     // Deconstructor
28     ~Point2d();
29
30     // Displays the values of _x, _y
31     // and addresses of the _prev and _next pointers
32     void display() const;
33
34     // Setter of _x
35     void setX(float);
36     // Getter of _x
37     float getX() const;
38
39     // Setter of _y
```

```
40      void setY(float);
41      // Getter of _y
42      float getY() const;
43
44      // Setter of _prev
45      void setPrev(Point2d*);
46      // Getter of _prev
47      Point2d* getPrev() const;
48
49      // Setter of _next
50      void setNext(Point2d*);
51      // Getter of _next
52      Point2d* getNext() const;
53
54      // Sets the value of _x and _y
55      void set(float, float);
56      // Sets the value of _x and _y with given another point by pointer
57      void set(const Point2d*);
58      // Sets the value of _x and _y with given another point by reference
59      void set(const Point2d&);
60
61      // Asks user to get _x and _y values
62      void askvalue();
63  };
64
65  #endif // POINT2D_H
```

1.2 Implement member function display(...) and operator overload

Listing 2: Point2d::display()

```
1  void Point2d::display() const
2  {
3      std::cout<<"POINT x:"<<_x<<" y:"<<_y;
4      std::cout<<" P:"<<std::hex<<_prev;
5      std::cout<<" N:"<<std::hex<<_next;
6      std::cout<<std::endl;
7  }
8
9  std::ostream& operator<<(std::ostream& os, const Point2d& point)
10 {
11   point.display();
12   return os;
13 }
14
15 std::ostream& operator<<(std::ostream& os, const Point2d* point)
16 {
17   point->display();
18   return os;
19 }
```

1.3 Setters, Getters and askvalue(...)

Listing 3: Setters-Getters

```
1  void Point2d::setX(float x)
2  {
3      _x = x;
4  }
5
6  float Point2d::getX() const
7  {
```

```
8       return _x;
9   }
10
11  void Point2d::setY(float y)
12  {
13      _y = y;
14  }
15
16  float Point2d::getY() const
17  {
18      return _y;
19  }
20
21  void Point2d::set(float x, float y)
22  {
23      _x = x;
24      _y = y;
25  }
26
27  void Point2d::set(const Point2d* point)
28  {
29      this->_x = point->getX();
30      this->_y = point->getY();
31  }
32
33  void Point2d::set(const Point2d& point)
34  {
35      this->_x = point.getX();
36      this->_y = point.getY();
37  }
38
39  void Point2d::askvalue()
40  {
41      std::cout<<"x?";
42      std::cin>>_x;
43      std::cout<<"y?";
44      std::cin>>_y;
45  }
```

1.4 Declare and Initialize a dummy Point2d

Listing 4: Dummy Test

```
1   int main(int argc, char *argv[])
2   {
3       cout<<"Point2d Dummy Test"<<endl;
4       cout<<"--------------------"<<endl;
5       Point2d *dummy = new Point2d();
6
7       dummy->display();
8       dummy->set(.4, 5.);
9       dummy->display();
10      dummy->setX(3.8);
11      dummy->setY(1.2);
12      dummy->display();
13      dummy->askvalue();
14      cout<<dummy;
15
16      delete dummy;
17      dummy = 0;
18      return 0;
19  }
```

2 Polygon

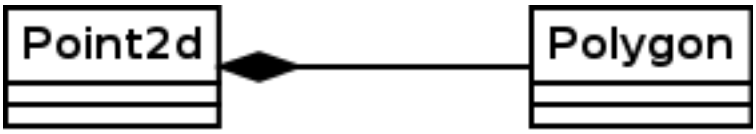


Figure 1: Polygon compose of Point2ds

2.1 Declare and implement Polygon class

Listing 5: Polygon.h

```
1  #ifndef POLYGON_H
2  #define POLYGON_H
3
4  #include "Point2d.h"
5
6  // Representing a polygon with 2D points
7  // Polygon class compose of Point2d classes
8  // Polygon class keeps root of the chain list
9  // represented with Point2d pointer: _start
10 class Polygon
11 {
12 private:
13     // Representing the first element in the double chained list
14     Point2d* _start;
15 public:
16     // Default constructor
17     Polygon();
18     // Desctructor
19     ~Polygon();
20
21     // Displays the indexes, addresses of the elements in chain list
22     // and displays Point2d itself
23     void display() const;
24
25     // Setter of the _start
26     void setStartPoint(Point2d*);
27     // Getter of the _start
28     Point2d* getStartPoint() const;
29
30     // Returns the first element in chain list: _start
31     Point2d* begin() const;
32     // Returns the size of the double chain list
33     int size() const;
34     // Returns the item in the given index
35     Point2d* get_item(int) const;
36     // Inserts the given item with pointer to the end of the chain list
37     void insert(Point2d*);
38     // Inserts the given item with pointer to the given index, otherwise end of the chain
           list
39     void insert_at(Point2d*, int = -1);
40     // Deletes the item from the given index
41     void delete_at(int);
42
43     // [] operator overload returns the item specified with index
44     Point2d* operator[](int);
45
46     // Creates new polygon and returns its pointer
47     static Polygon* BuildPolygon();
48     // Creates new polygon, creates number of Point2d elements specified with parameter
49     // Ask user for values of the points and returns the created polygon's pointer
```

```
50      static Polygon* BuildPolygon(int);
51      // Creates number of Point2d elements specified with parameters and asks user their
            values
52      // and assign them to the given polygon by pointer
53      static void BuildPolygon(Polygon*, int);
54      // Creates number of Point2d elements specified with parameters and asks user their
            values
55      // and assign them to the given polygon by reference
56      static void BuildPolygon(Polygon&, int);
57  };
58
59  #endif // POLYGON_H
```

2.2 Declare and Implement BuildPolygon(...) function

Listing 6: BuildPolygon()

```
1   Polygon* Polygon::BuildPolygon()
2   {
3       return new Polygon();
4   }
5
6   Polygon* Polygon::BuildPolygon(int nPoints)
7   {
8       Polygon* polygon = new Polygon();
9
10      for(int i = 0; i < nPoints; i++)
11      {
12          Point2d* point = new Point2d();
13
14          point->askvalue();
15
16          polygon->insert(point);
17      }
18
19      return polygon;
20  }
21
22  void Polygon::BuildPolygon(Polygon* polygon, int nPoints)
23  {
24      if(!polygon)
25          polygon = Polygon::BuildPolygon();
26
27      for(int i = 0; i < nPoints; i++)
28      {
29          Point2d* point = new Point2d();
30
31          point->askvalue();
32
33          polygon->insert(point);
34      }
35  }
36
37  void Polygon::BuildPolygon(Polygon& polygon, int nPoints)
38  {
39      for(int i = 0; i < nPoints; i++)
40      {
41          Point2d* point = new Point2d();
42
43          point->askvalue();
44
45          polygon.insert(point);
46      }
47  }
```

2.3 Declare and Implement function that displays elements of polygon

Listing 7: Polygon::display()

```
void Polygon::display() const
{
    std::cout<<"Polygon:"<<std::endl;

    for(int i = 0; i < this->size(); i++)
    {
        Point2d *point = this->get_item(i);
        std::cout<<"index:"<<i<<" addr:"<<std::hex<<point<<std::endl;
        point->display();
        point = 0;
    }
}

std::ostream& operator<<(std::ostream& os, const Polygon& polygon)
{
  polygon.display();
  return os;
}

std::ostream& operator<<(std::ostream& os, const Polygon* polygon)
{
  polygon->display();
  return os;
}
```

3 Insertion and deletion of elements

Here is the section for double chained list data structure functions implemented.

3.1 begin() that returns a pointer to the first element

Listing 8: Polygon::begin() const

```
Point2d* Polygon::begin() const
{
    return _start;
}
```

3.2 size() that returns the number of points in the polygon

Listing 9: Polygon::size() const

```
int Polygon::size() const
{
    int size = 0;

    Point2d* temp = this->begin();

    if(!temp) return 0;

    do
    {
        size++;
```

```
12          temp = temp->getNext();
13      }while(temp != this->begin());
14
15      return size;
16  }
```

3.3 getitem() that returns a pointer to a 2D Point at position in a given polygon

Listing 10: Polygon::getitem(int) const

```
1  Point2d* Polygon::get_item(int index) const
2  {
3      if(index >= this->size())
4      {
5          std::cerr<<"get_item(): Index out of range at index:"<<index<<" size was:"<<this
               ->size()<<std::endl;
6          return 0;
7      }
8
9      Point2d* temp = this->begin();
10
11     for(int i = 0; i < index; i++)
12         temp = temp->getNext();
13
14     return temp;
15 }
```

3.4 insertat() that inserts an element at a given position in the list

Listing 11: Polygon::insertat(int)

```
1  void Polygon::insert(Point2d* point)
2  {
3      Point2d* temp = this->begin();
4
5      if(!temp)
6      {
7          this->setStartPoint(point);
8          point->setNext(point);
9          point->setPrev(point);
10         return;
11     }
12
13     while(temp->getNext() != this->begin())
14         temp = temp->getNext();
15
16     point->setPrev(temp);
17     point->setNext(this->begin());
18     temp->setNext(point);
19     this->begin()->setPrev(point);
20 }
21
22 void Polygon::insert_at(Point2d* point, int index)
23 {
24     if(index == -1)
25         this->insert(point);
26
27     int size = this->size();
28
29     if(index > size || index < 0)
30     {
```

```
31          std::cerr<<"insert_at(): Index out of range at index:"<<index<<" size was:"<<this
                ->size()<<" to "<<point<<std::endl;
32          return;
33      }
34      if(index == size)
35          this->insert(point);
36
37
38      Point2d* temp = this->get_item(index);
39
40      point->setPrev(temp->getPrev());
41      point->setNext(temp);
42      temp->getPrev()->setNext(point);
43      temp->setPrev(point);
44  }
```

3.5 deleteat() that deletes (if possible) an element at given position

Listing 12: Polygon::deleteat(int)

```
1   void Polygon::delete_at(int index)
2   {
3       if(index >= this->size())
4       {
5           std::cerr<<"delete_at(): Index out of range at index:"<<index<<" size was:"<<this
                ->size()<<std::endl;
6           return;
7       }
8
9       Point2d* temp = this->get_item(index);
10
11      temp->getPrev()->setNext(temp->getNext());
12      temp->getNext()->setPrev(temp->getPrev());
13
14      delete temp;
15  }
```

3.6 Overload the operator [] for such class

Listing 13: Polygon::operator[

```
1   Point2d* Polygon::operator[](int index)
2   {
3       if(index >= this->size())
4       {
5           std::cerr<<"operator[](): Index out of range at index:"<<index<<" size was:"<<
                this->size()<<std::endl;
6           return 0;
7       }
8
9       Point2d* temp = this->get_item(index);
10
11      return temp;
12  }
```

4 Results

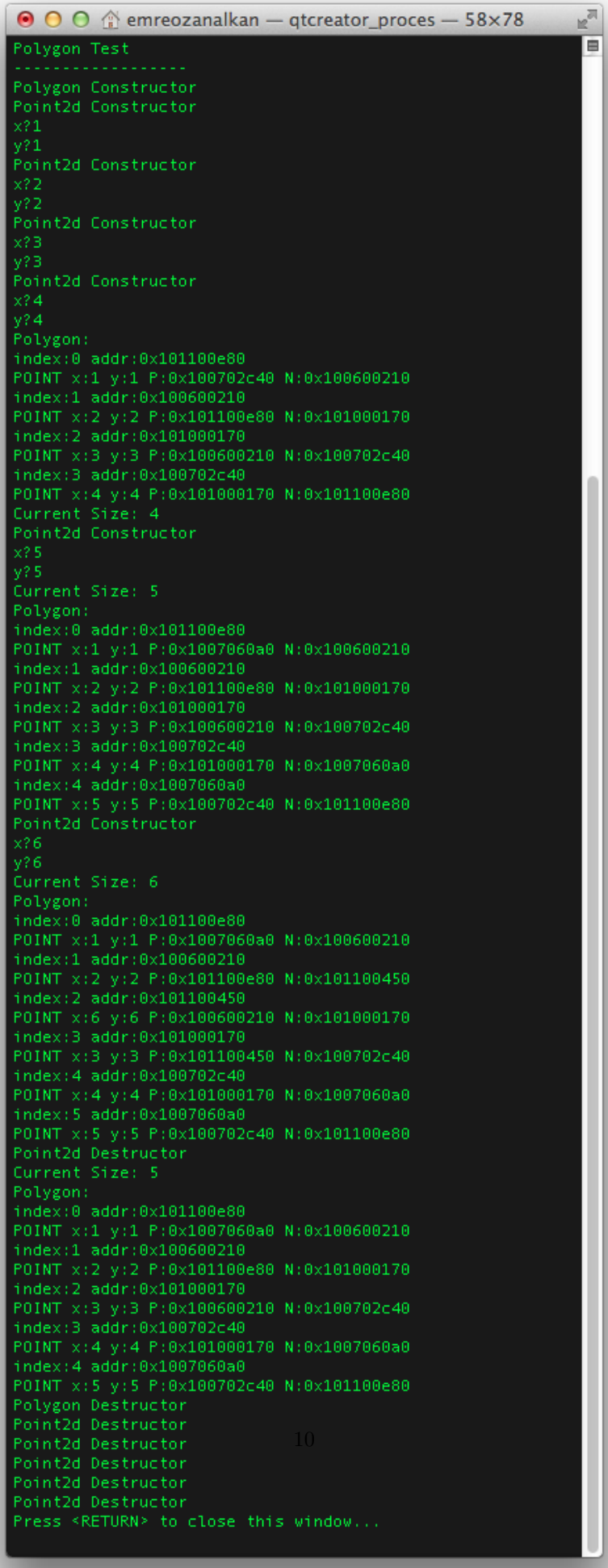
Results I obtained

4.1 Example main

Listing 14: main.cpp

```
1  int main(int argc, char *argv[])
2  {
3      cout<<"Point2d Dummy Test"<<endl;
4      cout<<"--------------------"<<endl;
5      Point2d *dummy = new Point2d();
6
7      dummy->display();
8      dummy->set(.4, 5.);
9      dummy->display();
10     dummy->setX(3.8);
11     dummy->setY(1.2);
12     dummy->display();
13     dummy->askvalue();
14     cout<<dummy;
15
16     delete dummy;
17     dummy = 0;
18
19     cout<<endl<<endl;
20
21
22     cout<<"Polygon Test"<<endl;
23     cout<<"--------------------"<<endl;
24
25     Polygon* polygon = Polygon::BuildPolygon(4);
26
27     polygon->display();
28
29     cout<<"Current Size: "<<polygon->size()<<endl;
30
31     Point2d* testPoint1 = new Point2d();
32     testPoint1->askvalue();
33
34     polygon->insert(testPoint1);
35
36     cout<<"Current Size: "<<polygon->size()<<endl;
37
38     polygon->display();
39
40     Point2d* testPoint2 = new Point2d();
41     testPoint2->askvalue();
42
43     polygon->insert_at(testPoint2, 2);
44
45     cout<<"Current Size: "<<polygon->size()<<endl;
46
47     polygon->display();
48
49     polygon->delete_at(2);
50
51     cout<<"Current Size: "<<polygon->size()<<endl;
52
53     cout<<polygon; //polygon->display();
54
55     delete polygon;
56
57     return EXIT_SUCCESS;
58 }
```

4.2 Example main


emreozanalkan — qtcreator_proces — 58×78
Polygon Test

Polygon Constructor
Point2d Constructor
x?1
y?1
Point2d Constructor
x?2
y?2
Point2d Constructor
x?3
y?3
Point2d Constructor
x?4
y?4
Polygon:
index:0 addr:0x101100e80
POINT x:1 y:1 P:0x100702c40 N:0x100600210
index:1 addr:0x100600210
POINT x:2 y:2 P:0x101100e80 N:0x101000170
index:2 addr:0x101000170
POINT x:3 y:3 P:0x100600210 N:0x100702c40
index:3 addr:0x100702c40
POINT x:4 y:4 P:0x101000170 N:0x101100e80
Current Size: 4
Point2d Constructor
x?5
y?5
Current Size: 5
Polygon:
index:0 addr:0x101100e80
POINT x:1 y:1 P:0x1007060a0 N:0x100600210
index:1 addr:0x100600210
POINT x:2 y:2 P:0x101100e80 N:0x101000170
index:2 addr:0x101000170
POINT x:3 y:3 P:0x100600210 N:0x100702c40
index:3 addr:0x100702c40
POINT x:4 y:4 P:0x101000170 N:0x1007060a0
index:4 addr:0x1007060a0
POINT x:5 y:5 P:0x100702c40 N:0x101100e80
Point2d Constructor
x?6
y?6
Current Size: 6
Polygon:
index:0 addr:0x101100e80
POINT x:1 y:1 P:0x1007060a0 N:0x100600210
index:1 addr:0x100600210
POINT x:2 y:2 P:0x101100e80 N:0x101100450
index:2 addr:0x101100450
POINT x:6 y:6 P:0x100600210 N:0x101000170
index:3 addr:0x101000170
POINT x:3 y:3 P:0x101100450 N:0x100702c40
index:4 addr:0x100702c40
POINT x:4 y:4 P:0x101000170 N:0x1007060a0
index:5 addr:0x1007060a0
POINT x:5 y:5 P:0x100702c40 N:0x101100e80
Point2d Destructor
Current Size: 5
Polygon:
index:0 addr:0x101100e80
POINT x:1 y:1 P:0x1007060a0 N:0x100600210
index:1 addr:0x100600210
POINT x:2 y:2 P:0x101100e80 N:0x101000170
index:2 addr:0x101000170
POINT x:3 y:3 P:0x100600210 N:0x100702c40
index:3 addr:0x100702c40
POINT x:4 y:4 P:0x101000170 N:0x1007060a0
index:4 addr:0x1007060a0
POINT x:5 y:5 P:0x100702c40 N:0x101100e80
Polygon Destructor
Point2d Destructor
Point2d Destructor
Point2d Destructor
Point2d Destructor
Point2d Destructor
Press <RETURN> to close this window...