LIST , Iterators and Algorithms

VGP131 – OOP II

INSTRUCTOR: IVALDO TRIBUTINO

Arrays

An array is a ***sequential structure*** meaning that it is a group of memory elements located in ***contiguous locations***. So we need to know how many elements there will be and then allocate a block of memory for them.

So we should start thinking about using another type of container if:

- We don't know how many elements there will be;
- If elements need to be inserted and deleted frequently.

ARRAY

Get Started | libraries.h | linkedList.h | main.cpp

main.cpp > main()

```
 7  * Ivaldo Tributino de Sousa <ISousa@lasallecollegevancouver.com>
 8  */
 9
10  #include "libraries.h"
11  // #include "linkedList.h"
12
13  using namespace std;
14
15  bool is_all_upper(const string& str);
16  bool findFrom(const string& str);
17
18  int main(){
19
20      //************************************************
21      //            -------- Array --------
22      //************************************************
23      cout << "-------- Array --------" << endl;
24
25
26      int a[] = {1,2,3,4,5,6,7,8,9,0};
27
28      for(int i=0; i < 10; i++){
29          cout << "Memory address of data " << a[i] << " is: " << &a[i]<<endl;
30      }
31
32
33
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
(base) Ivaldo:week3 admin$ ./main
-------- Array --------
Memory address of data 1 is: 0x7ffeef65a780
Memory address of data 2 is: 0x7ffeef65a784
Memory address of data 3 is: 0x7ffeef65a788
Memory address of data 4 is: 0x7ffeef65a78c
Memory address of data 5 is: 0x7ffeef65a790
Memory address of data 6 is: 0x7ffeef65a794
Memory address of data 7 is: 0x7ffeef65a798
Memory address of data 8 is: 0x7ffeef65a79c
Memory address of data 9 is: 0x7ffeef65a7a0
Memory address of data 0 is: 0x7ffeef65a7a4
(base) Ivaldo:week3 admin$
```

master*+ | 0 ⚠ 0 | Live Share | Ln 19, Col 1 | Spaces: 4 | UTF-8 | LF | C++ | Mac

Vector

Vectors are sequence containers that utilize continuous storage locations to store elements. They can manage storage and grow dynamically in an efficient way. These abilities come at a price: vectors consume more memory in exchange to handle storage and grow dynamically in size.

VECTOR

Get Started | libraries.h | linkedList.h | main.cpp

main.cpp > main()

```
35  //************************************************
36  cout << "-------- Vector --------" << endl;
37
38
39  vector<int> v(10);
40  copy(a, a+10, v.begin());
41
42  cout << "Vectors are assigned memory in blocks of contiguous locations" << endl;
43  for(int i=0; i < 10; i++){
44      cout << "Memory address of data " << v[i] << " is: " << &v[i]<<endl;
45  }
46
47  v.erase(v.begin()+4);
48
49  cout << "What will be the address of the elements after erose one of them?" << endl;
50
51  for(int i=0; i < 10; i++){
52      cout << "Memory address of data " << v[i] << " is: " << &v[i]<<endl;
53  }
54
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

```
Vectors are assigned memory in blocks of contiguous locations
Memory address of data 1 is: 0x7fe9d1d02680
Memory address of data 2 is: 0x7fe9d1d02684
Memory address of data 3 is: 0x7fe9d1d02688
Memory address of data 4 is: 0x7fe9d1d0268c
Memory address of data 5 is: 0x7fe9d1d02690
Memory address of data 6 is: 0x7fe9d1d02694
Memory address of data 7 is: 0x7fe9d1d02698
Memory address of data 8 is: 0x7fe9d1d0269c
Memory address of data 9 is: 0x7fe9d1d026a0
Memory address of data 0 is: 0x7fe9d1d026a4
What will be the address of the elements after erose one of them?
Memory address of data 1 is: 0x7fe9d1d02680
Memory address of data 2 is: 0x7fe9d1d02684
Memory address of data 3 is: 0x7fe9d1d02688
Memory address of data 4 is: 0x7fe9d1d0268c
Memory address of data 6 is: 0x7fe9d1d02690
Memory address of data 7 is: 0x7fe9d1d02694
Memory address of data 8 is: 0x7fe9d1d02698
Memory address of data 9 is: 0x7fe9d1d0269c
Memory address of data 0 is: 0x7fe9d1d026a0
Memory address of data 0 is: 0x7fe9d1d026a4
(base) Ivaldo:week3 admin$
```

master*+ 0 0 Live Share Ln 46, Col 1 Spaces: 4 UTF-8 LF C++ Mac

Vector vs. List

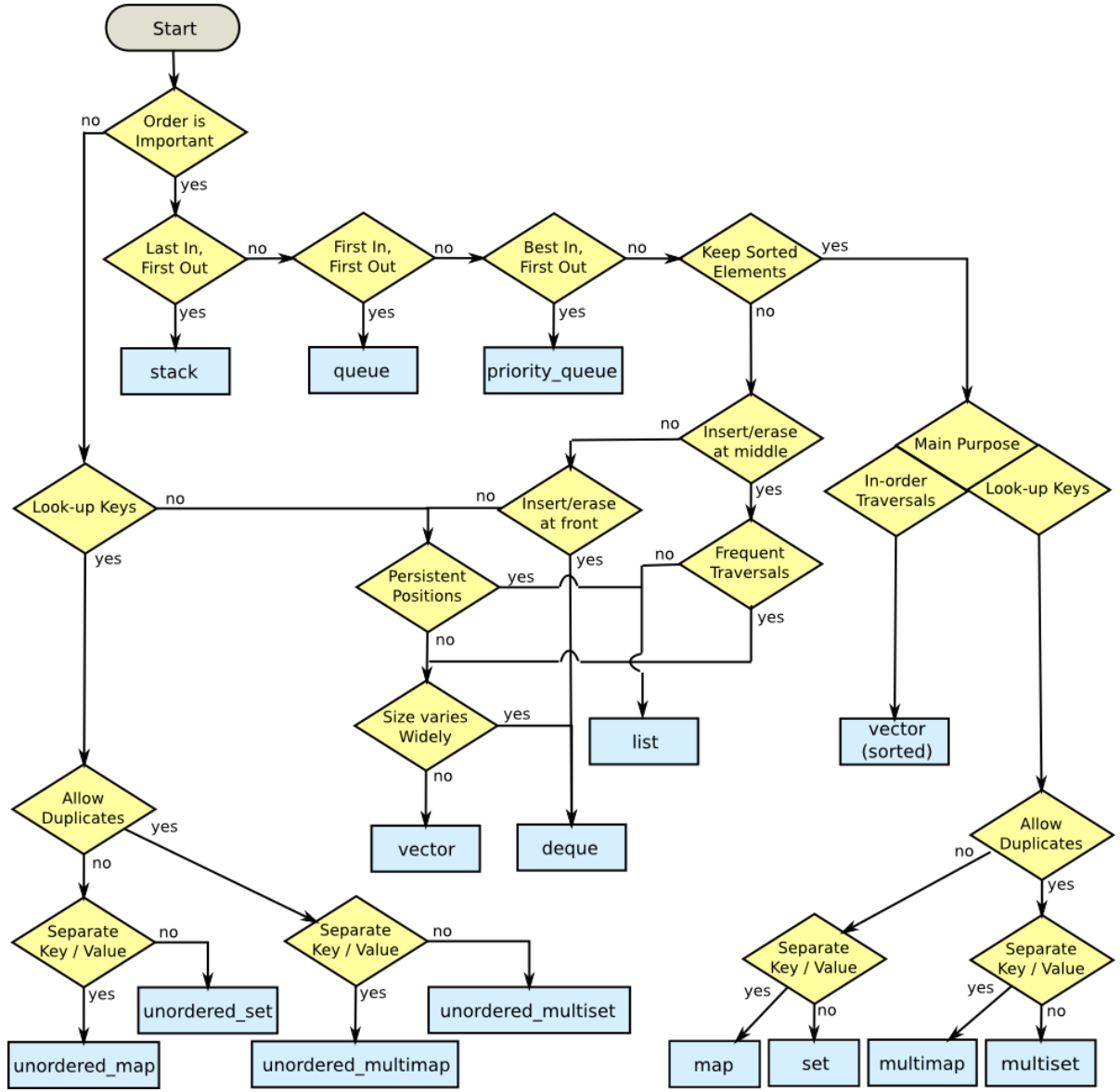
As the memory of the Vector is pre-allocated, in case of insertion if the memory becomes insufficient, new contiguous memory needs to be allocated and the elements are shifted in it whereas there is no issue of memory insufficiency in List as the memory is allocated dynamically.

One of the most important advantages of Vector is that it is thread-safe whereas the List in C++ is not thread-safe.

When talking about the memory efficiency, Vector is considered to be more effective as it needs memory for the element to be stored only whereas in the case of List, (implemented as a doubly-linked list) memory required to hold the single element is very large as the memory for the pointers to hold the address of preceding and succeeding node is also required.

As per the programmer's perspective, dealing with the pointers is tricky and requires higher-level coding knowledge (especially for the newbies) so working on the List is comparatively more difficult than the Vectors which deals with the normal array operations.

For a small number of elements, the vector is comparatively much faster than the List as the searching and the insertion becomes very easy and the array which is implemented in the Vector can be traversed and accessed easily.


Start
Order is Important
no
yes
Last In, First Out
First In, First Out
Best In, First Out
Keep Sorted Elements
stack
queue
priority_queue
Look-up Keys
Insert/erase at middle
Insert/erase at front
Persistent Positions
Frequent Traversals
Size varies Widely
list
vector
deque
Main Purpose
In-order Traversals
Look-up Keys
vector (sorted)
Allow Duplicates
Separate Key / Value
unordered_map
unordered_set
unordered_multimap
unordered_multiset
map
set
multimap
multiset

Advantage of Linked Lists.

Each element for a linked list can be stored anywhere in the free memory space - the elements in the list do not have to be stored in contiguous locations.

Each element in a linked list is referred to as a **node**. In a simple singly linked list, each element has two fields:

- Data member - contains the data
- Link member - contains the pointer to the next .

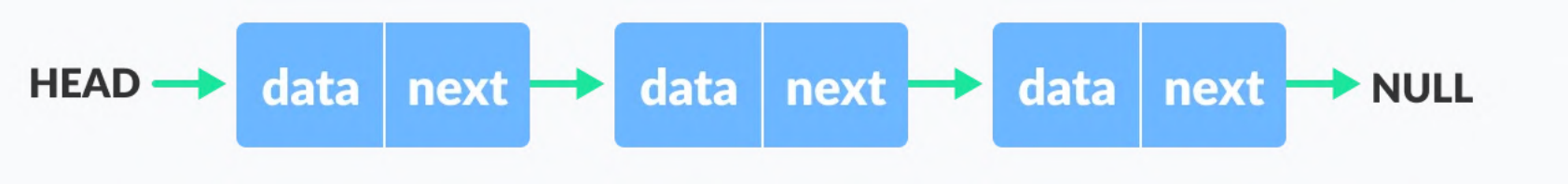


Image from [2]

LINKED LIST

Get Started | main.cpp | linkedList.h | Polygon.h | Polygon.cpp

linkedList.h > Node<T>

```
template<class T>
struct Node
{
    const T & data;
    Node *next;

    Node(const T & data);
    ~Node();
};

template <class T>
Node<T> :: Node(const T & data) : data(data), next(nullptr) { };

template <class T>
Node<T> :: ~Node<T>()
{
    std::cout << "Node_";
};

template <class T>
class linkedList
{
private:
    int count_;         //variable to store the number of elements in the list
    Node<T> *head_;    //pointer to the first node of the list
    Node<T> *thru_;     //pointer to traverse the list

public:

    linkedList();
    const T & operator[](unsigned index);
    void insertAtFront(const T & data);
    void display();
    int length() const;
    void insertPosition(unsigned index, const T & data);
    void deleteNote(const T & data);

    ~linkedList() {
        thru_ = nullptr;
```

master*+ | 0 ⚠ 0 | Live Share | Ln 12, Col 18 | Spaces: 4 | UTF-8 | LF | C++ | Mac

NODE

Get Started | main.cpp ● | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
55  //************************************************
56  //            -------- Node<T> --------
57  //************************************************
58  cout << "-------- Node<T> --------" << endl;
59  {
60  Node<int> n_1(20);
61  Node<int>* n_2 = new Node<int>(10);
62  n_1.next = n_2;
63  cout << n_1.data << endl;
64  cout << n_2 << endl;
65  cout << n_2->data << endl;
66  cout << n_1.next->data << endl;
67  delete n_2;
68  }
69  cout << '\n';
70  //************************************************
71  //            -------- LINKEDLIST --------
72  //************************************************
73  cout << "-------- LINKEDLIST --------" << endl;
74
75
76
77
78
79
80
81
82
83
84
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

```
-------- Node<T> --------
20
0x7fd406500000
10
10
Node_Node_
-------- LINKEDLIST --------
(base) Ivaldo:week3 admin$
```

0 ⚠ 0 Live Share

Ln 84, Col 5 Spaces: 4 UTF-8 LF C++ Mac

LINKED LIST

Get Started | main.cpp | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
58      cout << "-------- Node<T> --------" << endl;
59      {
60      Node<int> n_1(20);
61      Node<int>* n_2 = new Node<int>(10);
62      n_1.next = n_2;
63      cout << n_1.data << endl;
64      cout << n_2 << endl;
65      cout << n_2->data << endl;
66      cout << n_1.next->data << endl;
67      delete n_2;
68      }
69      cout << '\n';
70      //***********************************************
71      //            -------- LINKEDLIST --------
72      //***********************************************
73      cout << "-------- LINKEDLIST --------" << endl;
74      {
75      linkedList<int> list;
76      list.insertAtFront(10); //l[4]
77      list.insertAtFront(13); //l[3]
78      list.insertAtFront(16); //l[2]
79      list.insertAtFront(19); //l[1]
80      list.insertAtFront(21); //l[0]
81
82      list.display();
83
84      std::cout << "list[0] = " << list[0] << std::endl;
85
86      }
87
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
-------- LINKEDLIST --------
21 19 16 13 10
list[0] = 21
Node_0 has been deleted
Node_1 has been deleted
Node_2 has been deleted
Node_3 has been deleted
Node_4 has been deleted
(base) Ivaldo:week3 admin$
```

Live Share | Ln 84, Col 55 | Spaces: 4 | UTF-8 | LF | C++ | Mac

LINKED LIST

Get Started | main.cpp | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
91      list.insertPosition(0, 888);
92      list.display();
93
94      list.deleteNote(888);
95      list.display();
96      }
97
98      cout << "-------- LINKEDLIST of Polygons --------" << endl;
99
100     {
101         linkedList<Polygon> polyList;
102         Polygon triangle;
103         Polygon square(1,4);
104         Polygon pentagon(1,5);
105
106         polyList.insertAtFront(triangle);
107         polyList.insertAtFront(square);
108         polyList.insertAtFront(pentagon);
109
110         for(int i = 0; i<3; i++){
111             cout << polyList[i].shapeName() << endl;
112         }
113
114     }
115
116
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

```
-------- LINKEDLIST of Polygons --------
Default Constructor Invoked
Constructor Invoked
Constructor Invoked
pentagon
square
triangle
Polygon destroyed
Polygon destroyed
Polygon destroyed
Node_0 has been deleted
Node_1 has been deleted
Node_2 has been deleted
(base) Ivaldo:week3 admin$
```

0 0 Live Share Ln 73, Col 45 (8 selected) Spaces: 4 UTF-8 LF C++ Mac

INSERTPOSITION

Get Started | main.cpp | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
70  //*************************************************
71  //            -------- LINKEDLIST --------
72  //*************************************************
73  cout << "-------- LINKEDLIST --------" << endl;
74  {
75  linkedList<int> list;
76  list.insertAtFront(10); //l[4]
77  list.insertAtFront(13); //l[3]
78  list.insertAtFront(16); //l[2]
79  list.insertAtFront(19); //l[1]
80  list.insertAtFront(21); //l[0]
81
82  list.display();
83
84  std::cout << "list[0] = " << list[0] << std::endl;
85
86  list.insertPosition(0, 45);
87  list.insertPosition(2, 100);
88  list.insertPosition(100,888);
89  list.insertPosition(100,888);
90  list.insertPosition(100,888);
91  list.insertPosition(0, 888);
92  list.display();
93
94  }
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
-------- LINKEDLIST --------
21 19 16 13 10
list[0] = 21
888 45 21 100 19 16 13 10 888 888 888
Node_0 has been deleted
Node_1 has been deleted
Node_2 has been deleted
Node_3 has been deleted
Node_4 has been deleted
Node_5 has been deleted
Node_6 has been deleted
Node_7 has been deleted
Node_8 has been deleted
Node_9 has been deleted
Node_10 has been deleted
(base) Ivaldo:week3 admin$
```

Live Share | Ln 86, Col 5 | Spaces: 4 | UTF-8 | LF | C++ | Mac

DELETENOTE

Get Started | main.cpp | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
76      list.insertAtFront(10); //l[4]
77      list.insertAtFront(13); //l[3]
78      list.insertAtFront(16); //l[2]
79      list.insertAtFront(19); //l[1]
80      list.insertAtFront(21); //l[0]
81
82      list.display();
83
84      std::cout << "list[0] = " << list[0] << std::endl;
85
86      list.insertPosition(0, 45);
87      list.insertPosition(2, 100);
88      list.insertPosition(100,888);
89      list.insertPosition(100,888);
90      list.insertPosition(100,888);
91      list.insertPosition(0, 888);
92      list.display();
93
94      list.deleteNote(888);
95      list.display();
96      }
97
98
99
100
101
102
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

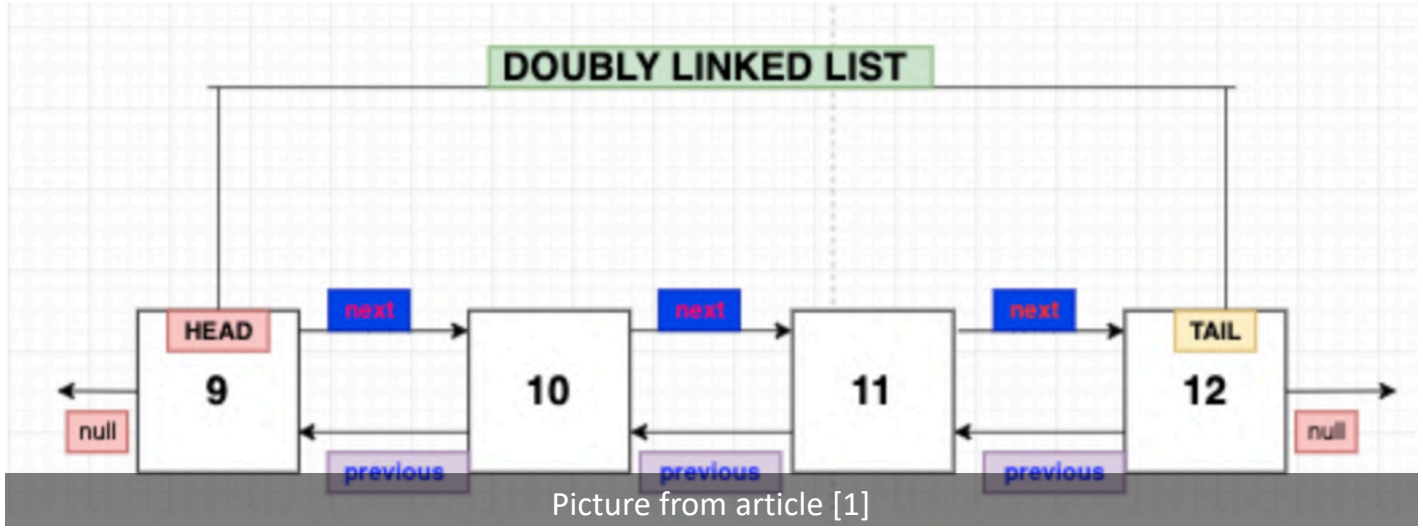
bash

```
888 45 21 100 19 16 13 10 888 888 888
Node_0 has been deleted
Node_7 has been deleted
Node_8 has been deleted
Node_9 has been deleted
45 21 100 19 16 13 10
Node_0 has been deleted
Node_1 has been deleted
Node_2 has been deleted
Node_3 has been deleted
Node_4 has been deleted
Node_5 has been deleted
Node_6 has been deleted
(base) Ivaldo:week3 admin$
```

0 ⚠ 0 | Live Share | Ln 95, Col 20 | Spaces: 4 | UTF-8 | LF | C++ | Mac

Sequence Container: list

- List containers are implemented as doubly linked lists.
- List is not a random access data structure, such as an array.
- To use list in a program, the program must include the following statement:
  #include <list>



Picture from article [1]

Various Ways to Declare a list Object

Statement	Description
list<elementType> listCont;	Creates the empty list container listCont. (The default constructor is invoked.)
list<elementType> listCont(otherList);	Creates the list container listCont and initializes it to the elements of otherList. listCont and otherList are of the same type.
list<elementType> listCont(size);	Creates the list container listCont of size size. listCont is initialized using the default constructor.
list<elementType> listCont(n, elm);	Creates the list container listCont of size n. listCont is initialized using n copies of the element elm.
list<elementType> listCont(beg, end);	Creates the list container listCont. listCont is initialized to the elements in the range [beg, end), that is, all the elements in the range beg...end-1. Both beg and end are iterators.

Various Operations Specific to a list Container

Expression	Description
listCont.assign(n, elem)	Assigns n copies of elem.
listCont.assign(beg, end)	Assigns all the elements in the range beg...end-1. Both beg and end are iterators.
listCont.push_front(elem)	Inserts elem at the beginning of listCont.
listCont.pop_front()	Removes the first element from listCont.
listCont.front()	Returns the first element. (Does not check whether the container is empty.)
listCont.back()	Returns the last element. (Does not check whether the container is empty.)
listCont.remove(elem)	Removes all the elements that are equal to elem.
listCont.remove_if(oper)	Removes all the elements for which oper is true.
listCont.unique(oper)	If the consecutive elements in listCont have the same value, removes the duplicates, for which oper is true.
listCont.sort(oper)	The elements of listCont are sorted. The sort criteria is specified by oper.

LIST

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
118     //*************************************************
119     cout << "-- List Container Example from our Textbook --" << endl;
120
121     list<int> intList1, intList2, intList3, intList4;
122
123     std::ostream_iterator<int> screen(cout, " ");
124
125     intList1.push_back(23);
126     intList1.push_back(58);
127     intList1.push_back(58);
128     intList1.push_back(58);
129     intList1.push_back(36);
130     intList1.push_back(15);
131     intList1.push_back(93);
132     intList1.push_back(98);
133     intList1.push_back(58);
134
135     cout << "Line 135: intList1: ";
136     copy(intList1.begin(), intList1.end(), screen);
137     cout << endl;
138
139     intList2 = intList1;
140
141     cout << "Line 141: intList2: ";
142     copy(intList2.begin(), intList2.end(), screen);
143     cout << endl;
144
145     intList1.unique();
146
147     cout << "Line 147: After removing the consecutive " << "duplicates," << endl
148     << " intList1: ";
149     copy(intList1.begin(), intList1.end(), screen);
150     cout << endl;
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
-- List Container Example from our Textbook --
Line 135: intList1: 23 58 58 58 36 15 93 98 58
Line 141: intList2: 23 58 58 58 36 15 93 98 58
Line 147: After removing the consecutive duplicates,
 intList1: 23 58 36 15 93 98 58
(base) Ivaldo:week3 admin$
```

Live Share | Ln 152, Col 1 (153 selected) | Spaces: 4 | UTF-8 | LF | C++ | Mac

LIST

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
146
147     cout << "Line 147: After removing the consecutive " << "duplicates," << endl
148     << " intList1: ";
149     copy(intList1.begin(), intList1.end(), screen);
150     cout << endl;
151
152     intList2.sort();
153
154     cout << "Line 154: After sorting, intList2: ";
155     copy(intList2.begin(),intList2.end(),screen);
156     cout << endl;
157
158     intList3.push_back(13);
159     intList3.push_back(25);
160     intList3.push_back(23);
161     intList3.push_back(198);
162     intList3.push_back(136);
163
164     cout << "Line 164: intList3: ";
165     copy(intList3.begin(), intList3.end(), screen);
166      cout << endl;
167
168     intList3.sort();
169
170     cout << "Line 170: After sorting, intList3: ";
171     copy(intList3.begin(), intList3.end(), screen);
172     cout << endl;
173
174     intList2.merge(intList3);
175
176     cout << "Line 176: After merging intList2 and " << "intList3, intList2: " << endl << " ";
177     copy(intList2.begin(), intList2.end(), screen); cout << endl;
178
```

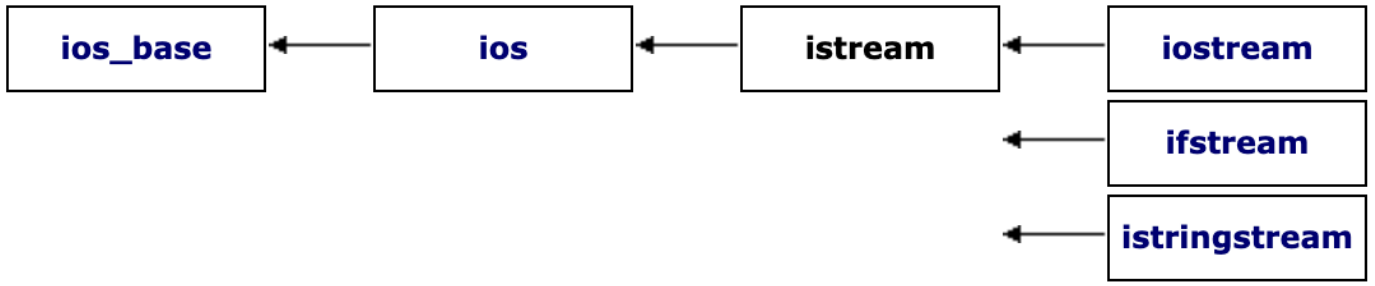
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

```
Line 154: After sorting, intList2: 15 23 36 58 58 58 58 93 98
Line 164: intList3: 13 25 23 198 136
Line 170: After sorting, intList3: 13 23 25 136 198
Line 176: After merging intList2 and intList3, intList2:
 13 15 23 23 25 36 58 58 58 58 93 98 136 198
(base) Ivaldo:week3 admin$
```

0 0 Live Share Ln 176, Col 22 Spaces: 4 UTF-8 LF C++ Mac

Iterators



Iterators are like pointers. In general, an iterator points to the elements of a container (sequence or associative). Thus, with the help of iterators, we can successively access each element of a container.

istream_iterator

The istream iterator is used to input data into a program from an input stream. The general syntax to use an istream iterator is:

```
istream_iterator<Type> isIdentifier(istream&);
```

where Type is either a built-in type or a user-defined class type, for which an input iterator is defined. The identifier **isIdentifier** is initialized using the constructor whose argument is either an **istream** class object such as **cin**, or any publicly defined **istream** subtype, such as **ifstream**. (To know more see [3] and [4].)

ostream_iterator *The ostream iterators are used to output data from a program to an output stream.*

The class ostream_iterator contains the definition of an output stream iterator. The general syntax to use an ostream iterator is:

```
ostream_iterator<Type> osIdentifier(ostream&);
```

or:

```
ostream_iterator<Type> osIdentifier(ostream&, char* deLimit);
```

where Type is either a built-in type or a user-defined class type, for which an output iterator is defined. The identifier **osIdentifier** is initialized using the constructor whose argument in either an **ostream** class object, such as **cout**, or any publicly defined **ostream** subtype, such as **ofstream**

ISTREAM & OSTREAM

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
180  //************************************************
181  //            -------- Iterators --------
182  //************************************************
183  cout << "-------- Iterators --------" << endl;
184
185  ifstream fileIn("francis.txt");
186
187  if(!fileIn.is_open())
188  {
189      cout << "Failed to open file!\n";
190      return 0;
191  }
192
193  std::istream_iterator< string > is(fileIn);
194  std::istream_iterator< string > eof;
195
196  vector< string > text;
197  copy( is, eof, back_inserter(text));
198
199
200  std::ostream_iterator<string> os(cout, " ");
201  copy( text.begin(), text.end(), os);
202  cout << endl;
203
204
205
206
207
208
209
210
211
212
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
-------- Iterators --------
Lord make Me an instrument of Your peace Where there is hatred let me sow love. Where there is injury, pardon. Where there is doubt, faith. Where there is despair, hope.
Where there is darkness, light. Where there is sadness joy. O Divine master grant that I may. Not so much seek to be consoled as to console. To be understood, as to under
stand. To be loved. as to love For it's in giving that we receive And it's in pardoning that we are pardoned And it's in dying that we are born To eternal life. Aaamen\n
}
(base) Ivaldo:week3 admin$
```

0 ⚠ 0 | Live Share | Ln 211, Col 5 | Spaces: 4 | UTF-8 | LF | C++ | Mac

ISTREAM & OSTREAM

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
179
180   //************************************************
181   //            -------- Iterators --------
182   //************************************************
183   cout << "-------- Iterators --------" << endl;
184
185   ifstream fileIn("francis.txt");
186
187   if(!fileIn.is_open())
188   {
189       cout << "Failed to open file!\n";
190       return 0;
191   }
192
193   // std::istream_iterator< string > is(fileIn);
194   // std::istream_iterator< string > eof;
195
196   // vector< string > text;
197   // copy( is, eof, back_inserter(text));
198
199   // std::ostream_iterator<string> os(cout, " ");
200   // copy( text.begin(), text.end(), os);
201   // cout << endl;
202
203   copy(std::istream_iterator< string >(fileIn), std::istream_iterator< string >(),std::ostream_iterator<string> (cout, " "));
204
205
206
207
208
209
210
211
212
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
Lord make Me an instrument of Your peace Where there is hatred let me sow love. Where there is injury, pardon. Where there is doubt, faith. Where there is despair, hope.
Where there is darkness, light. Where there is sadness joy. O Divine master grant that I may. Not so much seek to be consoled as to console. To be understood, as to under
stand. To be loved. as to love For it's in giving that we receive And it's in pardoning that we are pardoned And it's in dying that we are born To eternal life. Aaamen\n
(base) Ivaldo:week3 admin$
```

0 ⚠ 0 | Live Share | Ln 214, Col 5 | Spaces: 4 | UTF-8 | LF | C++ | Mac

CORRECTING YOUR TEXT

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
192
193     std::istream_iterator< string > is(fileIn);
194     std::istream_iterator< string > eof;
195
196     vector< string > text;
197     copy( is, eof, back_inserter(text));
198
199     std::ostream_iterator<string> os(cout, " ");
200     copy( text.begin(), text.end(), os);
201     cout << endl;
202
203     // copy(std::istream_iterator< string >(fileIn), std::istream_iterator< string >(),std::ostream_iterator<string> (cout, " "));
204
205     cout << "- Now let's remove the character " << text.back() << endl;
206
207     vector<string>::iterator lastElem;
208
209     text.erase(remove(text.begin(), text.end(), "}"));
210
211     copy(text.begin(), text.end(), os);
212     cout << endl;
213
214     cout << "- Finally, let replace the newline character with Full stop" << endl;
215
216     text.at(text.size()-1) = "AMEN.";
217
218     copy(text.begin(), text.end(), os);
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
-------- Iterators --------
Lord make Me an instrument of Your peace Where there is hatred let me sow love. Where there is injury, pardon. Where there is doubt, faith. Where there is despair, hope.
Where there is darkness, light. Where there is sadness joy. O Divine master grant that I may. Not so much seek to be consoled as to console. To be understood, as to under
stand. To be loved. as to love For it's in giving that we receive And it's in pardoning that we are pardoned And it's in dying that we are born To eternal life. Aaamen\n
}
```

0 0 Live Share | Ln 205, Col 16 | Spaces: 4 | UTF-8 | LF | C++ | Mac

Algorithms

Some of the operations are very specific to a container and, therefore, are provided as part of the container definition. However, several operations—such as find, sort, and merge—are common to all containers. These common operations are, therefore, provided as generic algorithms, and can be applied to all containers as well as the built-in array type.

The algorithms in the STL can be classified into the following categories:

- Nonmodifying algorithms
- Modifying algorithms
- Numeric algorithms

Most of the generic algorithms are contained in the header file algorithm. Certain algorithms, such as numeric, are contained in the header file numeric.

NONMODIFYING ALGORITHMS

Nonmodifying algorithms do not modify the elements of the container; they only investigate the elements.

Nonmodifying Algorithms		
adjacent_find	find_if	max
binary_search	find_end	max_element
count	find_first_of	min
count_if	for_each	min_element
equal	includes	search
equal_range	lower_bound	search_n
find	mismatch	upper_bound

Functions find and find_if

The functions find and find_if are used to find the elements in a given range. These functions are defined in the header file algorithm. The prototypes of the functions find and find_if are:

```
template <class inputItr, class size, class Type>
 inputItr find( inputItr first, inputItr last,
                const Type& searchValue);

template <class inputItr, class unaryPredicate>
inputItr find_if(inputItr first, inputItr last, unaryPredicate op);
```

The function find searches the range of elements **first. . .last-1** for the element **searchValue**. If searchValue is found in the range, the function returns the position in the range where searchValue is found; otherwise, it returns last.

The function find_if searches the range of elements **first. . .last-1** for the element for which op(rangeElement) is true. If an element satisfying op(rangeElement) true is found, it returns the position in the given range where such an element is found; otherwise, it returns last.

FIND & FIND_IF

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygo

main.cpp > main()

```cpp
//*************************************************
//         -------- Algorithms --------
//*************************************************
cout << "-------- Algorithms --------" << endl;

cout << "- Using Find" << endl;

vector<string>::iterator position;
position = find(text.begin(), text.end(), "darkness,");
copy(position, text.end(), os);
cout<< '\n';

cout << "- Using Find_if" << endl;

cout<< "1 = true and 0 = false, Result: " <<is_all_upper(text.back()) << endl;
position = find_if(text.begin(), text.end(), is_all_upper);
copy(text.begin(), position, os);
```

```cpp
bool is_all_upper(const string& str){

    bool flag = true; // for verification

    if(str.length() == 1) flag = false;

    for (int i = 0; i < str.size(); i++) // till string length
        if (!isupper(str[i]) && str[i]!= '.') {
            flag = false;
            break;
        }

    return flag;
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

```
Lord make Me an instrument of Your peace Where there is hatred let me sow love. Where there is injury, pardon. Where there is doubt, faith. Where there is despair, hope.
Where there is darkness, light. Where there is sadness joy. O Divine master grant that I may. Not so much seek to be consoled as to console. To be understood, as to under
stand. To be loved. as to love For it's in giving that we receive And it's in pardoning that we are pardoned And it's in dying that we are born To eternal life. AMEN.
-------- Algorithms --------
```

Live Share | Ln 235, Col 45 | Spaces: 4 | UTF-8 | LF | C++ | Mac

MODIFYING ALGORITHMS

Modifying algorithms, as the name implies, modify the elements of a container by rearranging, removing, and/or changing the values of the elements.

Modifying Algorithms		
copy	previous_permutation	rotate_copy
copy_backward	random_shuffle	set_difference
fill	remove	set_intersection
fill_n	remove_copy	set_symmetric_difference
generate	remove_copy_if	set_union
generate_n	remove_if	sort
inplace_merge	replace	stable_partition
merge	replace_copy_if	swap
next_permutation	replace_if	swap_ranges
partial_sort	reverse_copy	unique
partial_sort_copy	rotate	unique_copy

Functions fill and fill_n

The prototypes of these functions are:

```
template <class forwardItr, class Type>
void fill(forwardItr first, forwardItr last, const Type& value);

template <class forwardItr, class size, class Type>
void fill_n(forwardItr first, size n, const Type& value);
```

The first two parameters of the function **fill** are forward iterators that specify the starting and ending positions of the container; the third parameter is the filling element. The first parameter of the function **fill_n** is a forward iterator that specifies the starting position of the container; the second parameter specifies the number of elements to be filled; and the third parameter specifies the filling element.

FILL & FILL_N

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
239
240     cout << "- Using fill and fill_n" << endl;
241
242     fill(text.end()-1,text.end(), "Amen." );
243     copy(text.begin(), text.end(), os);
244     cout << "\n";
245
246     text.resize(text.size()+3);
247     fill_n(text.rbegin(), 3, "!");
248     copy(text.begin(), text.end(), os);
249     cout << "\n";
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

0 ⚠ 0 Live Share

Ln 267, Col 1 Spaces: 4 UTF-8 LF C++ Mac

NUMERIC ALGORITHMS

Numeric algorithms are designed to perform numeric calculations on the elements of a container.

Numeric Algorithms	
accumulate	inner_product
adjacent_difference	partial_sum

REMOVING DUPLICATES

Get Started | main.cpp ● | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
251     //************************************************
252     //  -------- REMOVING DUPLICATES --------
253     //************************************************
254     cout << "-------- REMOVING DUPLICATES --------" << endl;
255     vector<string> emails;
256     ifstream isFile("mbox-short.txt");
257     string str;
258     while(getline(isFile, str)) // storing into str until the delimitation character '\n' is find
259     {
260         if (str.find("From:") != string::npos)
261         {
262             int pos = str.find(" ");
263
264             emails.push_back(str.substr(pos+1));
265             cout << str.substr(pos+1) << endl;
266
267         };
268     }
269
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

bash

```
 !
-------- REMOVING DUPLICATES --------
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
zqian@umich.edu
rjlowe@iupui.edu
zqian@umich.edu
rjlowe@iupui.edu
cwen@iupui.edu
cwen@iupui.edu
gsilver@umich.edu
gsilver@umich.edu
zqian@umich.edu
gsilver@umich.edu
wagnermr@iupui.edu
zqian@umich.edu
antranig@caret.cam.ac.uk
gopal.ramasammycook@gmail.com
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
david.horwitz@uct.ac.za
stephen.marquard@uct.ac.za
louis@media.berkeley.edu
louis@media.berkeley.edu
```

REMOVING DUPLICATES

Get Started | main.cpp | libraries.h | linkedList.h | Polygon.h | Polygon.cpp

main.cpp > main()

```
258     while(getline(isFile, str)) // storing into str until the delimitation character '\n' is find
259     {
260         if (str.find("From:") != string::npos)
261         {
262             int pos = str.find(" ");
263 
264             emails.push_back(str.substr(pos+1));
265             cout << str.substr(pos+1) << endl;
266 
267         };
268     }
269 
270     cout << "- After removing the duplication" << endl;
271 
272     std::sort(emails.begin(), emails.end());
273     emails.erase(std::unique(emails.begin(),emails.end()), emails.end());
274 
275     vector<string> :: iterator itr;
276 
277     for(itr = emails.begin(); itr != emails.end(); ++itr){
278         cout << *itr << endl;
279     }
280 
281 
282    return 0;
283 }
284 
285 bool is_all_upper(const string& str){
```

PROBLEMS | OUTPUT | TERMINAL | DEBUG CONSOLE

bash

```
- After removing the duplication
antranig@caret.cam.ac.uk
cwen@iupui.edu
david.horwitz@uct.ac.za
gopal.ramasammycook@gmail.com
gsilver@umich.edu
louis@media.berkeley.edu
ray@media.berkeley.edu
rjlowe@iupui.edu
stephen.marquard@uct.ac.za
wagnermr@iupui.edu
zqian@umich.edu
(base) Ivaldo:week3 admin$
```

Live Share | Ln 270, Col 46 | Spaces: 4 | UTF-8 | LF | C++ | Mac

References

1) Soner Mezgitci (2022/01/15) https://mezgitci9.medium.com/difference-between-singly-linked-list-and-doubly-linked-list-in-javascript-99d7f36d0af
2) Programiz (2022/01/15) Linked list Data Structure, https://www.programiz.com/dsa/linked-list
3) C++(2022/01/15) std::istream, https://www.cplusplus.com/reference/istream/istream/
4) C++(2022/01/15) std::ifstream, https://www.cplusplus.com/reference/fstream/ifstream/
5) BOOK : C++ Programming From Problem Analysis to Program Design - D. S. Malik – 2011.
6) EDUCBA(2022/01/14) C++ vector vs list, https://www.educba.com/c-plus-plus-vector-vs-list/

Appendix

- Libraries.h
- linkedList.h
- Polygon.h
- Polygon.cpp
- main.cpp