

```

1  #include <iostream>
2  using namespace std;
3
4  // implementing the dynamic List ADT using array
5  // operations to be implemented: read, Modify, delete, isEmpty, insert, countElements
6
7  class List{
8
9      private:
10         int *array;
11         int maxSize; // useful to decide if resizing (doubling the array size) is needed
12         int endOfArray;
13
14     public:
15         List(int size){
16             array = new int[size];
17             maxSize = size;
18             endOfArray = -1;
19         }
20
21         bool isEmpty(){
22
23             if (endOfArray == -1)
24                 return true;
25
26             return false;
27         }
28
29         void resize(int s){
30
31             int *tempArray = array;
32
33             array = new int[s];
34
35             for (int index = 0; index < min(s, endOfArray+1); index++){
36                 array[index] = tempArray[index];
37             }
38
39             maxSize = s;
40
41         }
42
43
44         void insert(int data){
45
46             if (endOfArray == maxSize-1)
47                 resize(2*maxSize);
48
49             array[++endOfArray] = data;
50
51         }
52
53
54         void insertAtIndex(int insertIndex, int data){
55
56             // if the user enters an invalid insertIndex, the element is
57             // appended to the array, after the current last element
58             if (insertIndex > endOfArray+1)
59                 insertIndex = endOfArray+1;
60
61             if (endOfArray == maxSize-1)
62                 resize(2*maxSize);
63
64             for (int index = endOfArray; index >= insertIndex; index--)
```

```

65         array[index+1] = array[index];
66
67         array[insertIndex] = data;
68         endOfArray++;
69     }
70
71
72
73     int read(int index){
74         return array[index];
75     }
76
77     void modifyElement(int index, int data){
78         array[index] = data;
79     }
80
81
82     void deleteElement(int deleteIndex){
83
84         // shift elements one cell to the left starting from deleteIndex+1 to
85         // endOfArray-1
86         // i.e., move element at deleteIndex + 1 to deleteIndex and so on
87         for (int index = deleteIndex; index < endOfArray; index++)
88             array[index] = array[index+1];
89
90         endOfArray--;
91     }
92
93
94     int countList(){
95         int count = 0;
96         for (int index = 0; index <= endOfArray; index++)
97             count++;
98
99         return count;
100    }
101
102    void print(){
103
104        for (int index = 0; index <= endOfArray; index++)
105            cout << array[index] << " ";
106
107        cout << endl;
108    }
109
110
111 };
112
113 int main(){
114     int listSize;
115
116     cout << "Enter list size: ";
117     cin >> listSize;
118
119     List integerList(1); // we will set the maxSize to 1 and double it as and when needed
120
121     for (int i = 0; i < listSize; i++){
122         int value;
123         cout << "Enter element # " << i << " : ";
124         cin >> value;
125
126
127

```

```

128         integerList.insertAtIndex(i, value);
129     }
130
131     integerList.print();
132
133
134     // to read an element at a particular index (before delete)
135
136     int readIndex;
137     cout << "Enter an index to read (before delete): ";
138     cin >> readIndex;
139     cout << "Value at " << readIndex << " is: " << integerList.read(readIndex) << endl;
140
141     // to delete an element at a particular index
142
143     int deleteIndex;
144     cout << "Enter an index to delete: ";
145     cin >> deleteIndex;
146     integerList.deleteElement(deleteIndex);
147
148     cout << "Contents of the List: ";
149     integerList.print();
150
151
152     // to read an element at a particular index (after delete)
153
154     cout << "Enter an index to read (after delete): ";
155     cin >> readIndex;
156     cout << "Value at " << readIndex << " is: " << integerList.read(readIndex) << endl;
157
158     cout << "Number of elements in the list (before insert) is: " << integerList.
countList() << endl;
159
160
161     // to insert an element at a particular index
162     int insertIndex, insertValue;
163     cout << "Enter an index to insert: ";
164     cin >> insertIndex;
165     cout << "Enter a value to insert: ";
166     cin >> insertValue;
167     integerList.insertAtIndex(insertIndex, insertValue);
168
169     cout << "Contents of the List: ";
170     integerList.print();
171
172     // to read an element at a particular index (after insert)
173
174     cout << "Enter an index to read (after insert): ";
175     cin >> readIndex;
176     cout << "Value at " << readIndex << " is: " << integerList.read(readIndex) << endl;
177
178     cout << "Number of elements in the list (after insert) is: " << integerList.
countList() << endl;
179
180     // to insert at the end of the list
181     cout << "Enter the element you want to insert at the end of the list: ";
182     cin >> insertValue;
183     integerList.insert(insertValue);
184
185     cout << "Contents of the List: ";
186     integerList.print();
187
188
189     return 0;

```

```
Enter list size: 5
Enter element # 0 : 99
Enter element # 1 : 88
Enter element # 2 : 22
Enter element # 3 : 11
Enter element # 4 : 33
99 88 22 11 33
Enter an index to read (before delete): 3
Value at 3 is: 11
Enter an index to delete: 3
Contents of the List: 99 88 22 33
Enter an index to read (after delete): 0
Value at 0 is: 99
Number of elements in the list (before insert) is: 4
Enter an index to insert: 0
Enter a value to insert: 11
Contents of the List: 11 99 88 22 33
Enter an index to read (after insert): 2
Value at 2 is: 88
Number of elements in the list (after insert) is: 5
Enter the element you want to insert at the end of the list: 44
Contents of the List: 11 99 88 22 33 44
```