Array Based Implementation of ADT List

CS 110C

Anita Rathi

Array List

- Look at the following codes to understand the Array Based implementation of ADT list.
  - ListInterface.h
  - ArrayList.cpp

Data Members

```
template <class ItemType>
class List{
  private :
    static const int SIZE= 100;// Default capacity of the list
    ItemType list[SIZE + 1];// Array of list items (ignore items[0])
    int itemCount;// Current count of list items
    int maxItems;// Maximum capacity of the list
```

Constructor and Destructor

```
public:
  List(){ // default constructor
    itemCount=0;
  }
  ~List(){ //default destructor
    itemCount=0;
  }
```

List Member Functions

```
bool isEmpty(){
   return(itemCount==0);
}

int getLength(){
   return itemCount;
}
```

List Member Functions

```
bool insert(int pos, const ItemType& item){
  if (pos>=1 && pos<= getLength()+1){ //position is valid
    if (getLength()+1 <=maxItems){ //scope of insertion is there
      //shifting the elements
      itemCount++;
      for(int i=itemCount; i>pos;i--)
        list[i]=list[i-1];
      list[pos]=item; //insertion
      cout<<"Successfully inserted "<<item <<endl;
    }
    else{
      cout<<"Item cannot be inserted"<<endl;
    }
  }
  else{
    cout<<"Position is invalid";
  }
}
```

List Member Functions

```
bool remove(int pos){
  if (pos>=1 && pos<= getLength()){ //position is valid
      //shifting the elements
      ItemType temp=list[pos];
      for(int i=pos;i<=getLength()-1; i++)
        list[i]=list[i+1];

      itemCount--; //reducing the number of elements by 1
      cout<<"Successfully deleted "<<temp<<endl;
  }
  else{
    cout<<"Item cannot be deleted"<<endl;
  }
}
```

List Member Functions

```
void clear(){
  itemCount=0;
  cout<<"All elements deleted";
}

ItemType getEntry(int pos) const{
  if(pos>=1 && pos<=itemCount)
    return(list[pos]);
  else
    throw logic_error("Invalid position");
}
```

List Member Functions

```
  ItemType getEntry(int pos) const{
    if(pos>=1 && pos<=itemCount)
      return(list[pos]);
    else
      throw logic_error("Invalid position");
  }

  void setEntry(int pos, const ItemType& item){
    if(pos>=1 && pos<=itemCount){
      list[pos]=item;
      cout<<"Successfully replaced item at "<< pos <<" with "<<item<<endl;
    }

    else
      cout<<"Invalid position";
  }
};
```

main()

```
int main(){
  List<int> L;
  L.insert(1,23);
  L.insert(2,50);
  L.insert(3,100);

  for(int i = 1; i<=L.getLength();i++)
    cout<< L.getEntry(i)<<" ";
  cout<<endl;
  L.remove(2);
  for(int i = 1; i<=L.getLength();i++)
    cout<< L.getEntry(i)<<" ";
  cout<<endl;
  L.insert(3,500);
  L.insert(4,30);
  for(int i = 1; i<=L.getLength();i++)
    cout<< L.getEntry(i)<<" ";
  cout<<endl;
  L.setEntry(3,1000);
  for(int i = 1; i<=L.getLength();i++)
    cout<< L.getEntry(i)<<" ";
  cout<<endl;
  L.clear();
  if(L.isEmpty())
    cout<<"List is empty"<<endl;
  else
    cout<<"List is not empty"<<endl;
}
```

Sample Output

```
Successfully inserted 23
Successfully inserted 50
Successfully inserted 100
23 50 100
Successfully deleted 50
23 100
Successfully inserted 500
Successfully inserted 30
23 100 500 30
Successfully replaced item at 3 with 1000
23 100 1000 30
All elements deletedList is empty
```