



11/10/2021

Department Of Computer Science

Subject Instructor: Ma'am Zainab Malik

Assignment: 2

Date: 10-11-2021

Class: BSCS-3B

Submitted by:

Madina Javed Iqbal Khokhar 2426

Task

Assigned Task:

Create a C++ based grocery application in which list of items along with their unique item ID, name, description and price per item will be displayed on screen for the user to add/delete items of his/choice in/from a cart (cart will be generated as a singular linked list). The application should behave as follow:

- System should load item ID, name, description and price per item from a text file in to a linked list (items) and then it should display all loaded details on screen.
- After that user should be provided with four options, i.e. 0,1,2,3 in an iterative manner.
- If user presses 1, the system should ask user to enter item ID and quantity and should add it into the cart.
- If user presses 2, the system should ask for item ID and then it should delete that item from the cart.
- If user presses 3, list of items with item ID, name, description and price per item should be displayed again.
- If user presses 0, the system should generate bill containing selected item, quantity, price per item and total amount to pay.

Node Class:

```
#include<iostream>
using namespace std;
template<class T> // Templates actually
//increase flexibility, they're easy // to update, and they provide
consistency
class Node
{
private:
T info; // variable name use to store information
Node<T> *next; // variable use to store address of next node,that's why its
data type

public:
Node(T i=0,Node<T> *n=0):info(i),next(n)
{
```

```

        // constructor { having
    }

void setInfo(T i); // using setter and getter
T getInfo();
void setNext(Node<T> *n); // using setter and getter
Node<T>* getNext();
}; //EOC

template<class T>
void Node<T>::setInfo(T i)
{
    info=i; // setting our info
}

template<class T>
void Node<T>::setNext(Node<T> *n)
{
    next=n; // setting our next
}

template<class T>
T Node<T>::getInfo() // getting our info
{
    return info;
}

template<class T>
Node<T>* Node<T>::getNext() // getting our next
{
    return next;
}

```

LinkedList Class:

```

#include <iostream>
#include "Node.h" // including Node.h file
#include<stdlib.h>
using namespace std;
template<class T>
class LinkedList
{
private:
    Node<T> *head; // head is variable whose data type is node
    Node<T> *tail; // tail is variable whose data type is node
public:
    LinkedList()

```

```

{
head=0; //constructor { having same name as class}
tail=0;
}
// here, we are declaring each functions to perform a particular task
void addToHead(T element);
void traversing();
T removeFromHead();          //it will delete first Node
void removeFromTail();
void displayBill();
void Remove(T element);
void variabletype();
Node<T>*getHead();
    double searching(Node<T>* element);
    Node<T>*setHead(Node<T>*u);
};

template<class T>
void LinkedList<T>::addToHead(T element){

Node<T> *ptr=new Node<T>(element);
if(head==0 && tail==0)          // for the first time , it will work, as head and
tail are equal to 0
{
head=ptr;
tail=ptr;
}
else          //only one element or >1 element
{
ptr->setNext(head);
head=ptr;
}

}          //addToHead

template<class T>
void LinkedList<T>::traversing(){
Node<T> *ptr=head;
while(ptr!=0) // till the end
{
    ptr->getInfo()->displayInfo();// getting the info of ptr and then
display
ptr=ptr->getNext(); // set ptr to next
}
}          //traversing

template<class T>

```

```

T LinkedList<T>::removeFromHead() // fuction use to remove from head
{
if(head==0) // as list is empty and there is no value where we cannot
delete any info
    // so error will occur

{
cerr<<"nothing to delete"<<endl;
}
else if(head==tail) // in case there is only one element exist
{
T info=head->getInfo();
delete head;
head=0;
tail=0; // after deletig , we will set head and tail equal to 0
return info;
}
else //More than one element
{
Node<T> *temp=head;
head=head->getNext();
T info=temp->getInfo();
delete temp;
return info;
}} //Remove From Head

template<class T>
void LinkedList<T>::removeFromTail() {
    if(head==0){
        cerr<<"List cannot be empty";
    }
    else if(head==tail){
        head=0;
        tail=0;
    }
    else{
        Node<T>*ptr=head; // storing head in temp
        Node<T>*beforeadd;
        while(ptr->getNext()!=0&&ptr->getInfo()!=tail->getInfo()){
            beforeadd=ptr;
            ptr=ptr->getNext();
        }

        delete tail;
        tail=beforeadd;
    }
}

```

```

}                                //Remove From Tail

template<class T>
void LinkedList<T>::displayBill(){
    Node<T>*ptr=head; // setting ptr to head
    while(ptr!=0){ // till the end
        cout<<ptr->getInfo()->getQuan()<<" ";
        cout<<ptr->getInfo()->getId()<<" ";
        cout<<endl;
        ptr=ptr->getNext(); // setting ptr to next
    }
} //displayBill

template<class T>
void LinkedList<T>::Remove(T element){
    if(head==0) //as list is empty and there is no value which we can delete
        // so error will occur

    {
        cerr<<"Nothing to delete"<<endl;
    }
    else if(head==tail && head->getInfo()->getId()==element->getId())
    {
        delete head;
        head=tail=0;
    }
    else if(head->getInfo()->getId()==element->getId())
    {
        cout<<removeFromHead()<<endl;
        cout<<"Value deleted";
    }
    else if(tail->getInfo()->getId()==element->getId())
    {
        removeFromTail(); // calling remove from head function
    }
    else
    {
        Node<T> *temp=head;
        while(temp!=tail && temp->getNext()->getInfo()->getId()!=element->getId())
        {
            temp=temp->getNext();
        }
        if(temp==tail)
        {
            cerr<<"Element not found"<<endl;
        }
    }
}

```

```

else
{
Node<T> *ptr=temp->getNext();
temp->setNext(ptr->getNext());
delete ptr;
}
}
}
//remove
template<class T>
double LinkedList<T>::searching(Node<T>* element)
{
//double p=element->getInfo()->getQuan();
Node<T> *ptr=head;
while(ptr!=0)
{
if(ptr->getInfo()->getId()==element->getInfo()->getId())
{
string str=ptr->getInfo()->getPrice();
cout<<ptr->getInfo()->getId()<<" ";
cout<<ptr->getInfo()->getName()<<" ";
cout<<ptr->getInfo()->getDes()<<" ";
double u=atof(str.c_str());
return p*u;
}
ptr=ptr->getNext();
}
cerr<<"item not found";
}
template <class T>
Node<T>* LinkedList<T>::getHead(){
return head;
}
template <class T>
Node<T>* LinkedList<T>::setHead(Node<T>*u){
    head=u;
}
}

```

Item Class:

```

#include<iostream>
using namespace std;
class items

```



```

{
private:
string itemName,uniqueId,description,price; // declaring variables
double quantity;
public: // constructor
items(string name="",string id="",string des="",string pri="")
{
itemName=name;// passing values
uniqueId=id;
description=des;
price=pri;
} // declaring functions
void setquan(double q);
double getquan();
void setId(string id);
string getId();
void displayInfo();
void setDes(string descr);
string getDes();
void setPrice(string pri);
string getPrice();
void setName(string name);
string getName();
};

void items::displayInfo(){ // displaying info
cout<<itemName<<" "<<uniqueId<<" "<<description<<"
"<<price<<endl;
}
void items::setquan(double q){ // setting quantity
    quantity=q;
}
void items::setId(string id){ // setting id
    uniqueId=id;
}

```

```

void items::setDes(string descr){ // setting description
    description=descr;
}
void items::setPrice(string pri){ // setting price
    price=pri;
}
void items::setName(string name){ // setting name of product
    itemName=name;
}
string items::getDes(){ // getting description
    return description;
}
string items::getPrice(){ // getting price
    return price;
}
string items::getName(){ // getting name
    return itemName;
}
string items::getId(){ // getting id
    return uniqueId;
}
double items::getquan(){ // getting quantity
    return quantity;
}

```

Main Class:

```

#include <iostream>
#include "LinkedList.h" // including linkedlist file
#include "items.h" // including item file
#include<fstream> // for file handling
#include<string> // to handle string
using namespace std;
string strings[4];

```

```

int len(string str){
    int length = 0;
    for (int i = 0; str[i] != '\0'; i++)
    {
        length++;
    }
    return length;
}

void split (string str, char seperator) {
    int currIndex = 0, i = 0;
    int startIndex = 0, endIndex = 0;
    while (i <= len(str)) // less than or equal to length of string
    {
        if (str[i] == seperator || i == len(str))
        {
            endIndex = i;
            string subStr = "";
            subStr.append(str, startIndex, endIndex - startIndex);
            strings[currIndex] = subStr;
            currIndex += 1;
            startIndex = endIndex + 1;
        }
        i++;
    }
}

int main() {

```

```

LinkedList<items*> item;// object
ifstream obj;
obj.open("Downloads"); // giving path
if(!obj){ // in case file does not exist
cout<<"File dont exist";
return 1;
}
else{
string str;
while(getline(obj,str)){
split(str,' ');
items *p1=new items(strings[0],strings[1],strings[2],strings[3]);
item.addToHead(p1); // calling add to head function
}
//displaying all the data
cout<<"Following are the grocery items"<<endl;
item.traversing(); // displaying items
LinkedList <items*> cart;
do{
int opt;
cout<<"Press 1 to enter an item in cart"<<endl;
cout<<"Press 2 to delete an item"<<endl;
cout<<"press 3 to display all the elements"<<endl;
cout<<"Press 4 to print bill"<<endl;
cout<<"enter the option ";
cin>>opt;
string id;
double quantity,totalprice;

```

```

items *p=new items();
switch(opt){

    case 0:
        while(cart.getHead()!=0)
        {
            double price;
            price=item.searching(cart.getHead());
            //cout<<cart.getHead()->getInfo()->getQuan()<<" ";
            cout<<price<<endl;
            cart.setHead(cart.getHead()->getNext());
            totalprice+=price;
        }
        cout<<"total Price: " <<totalprice;
    exit(-1);
    break;

    case 1:
        cout<<"enter the Id: ";
        cin>>id;
        cout<<"Enter quantity: ";
        cin>>quantity;
        p->setId(id);
        p->setQuan(quantity);
        cart.addToHead(p);
        break;

    case 2:

```

```
cout<<"Enter the Id: ";  
cin>>id;  
p->setId(id);  
cart.Remove(p);  
break;
```

case 3:

```
cout<<"Grocery items"<<endl;  
item.traversing(); // calling traversing function to display items  
break;
```

```
}  
}while(true);  
}  
}
```

Output:

C:\Users\ACER\Desktop\BSCS\program\Assignment 2.exe

file dont exist

process exited after 0.07784 seconds with return value 1

press any key to continue . . .