Name : Mark Yean Tuck Ming

TP Number : TP044716

Intake Code :UC2F1902IS

Handout date : 19 August 2019

Submission date : 21 October 2019

Introduction

This program is designed as a solution to customer service personnel which handle all the primary information of the customer orders. This food order and delivery system should be able to record a new food order, customer recipient information and able to display record as well as modify the record.

One of the functions of this program is to store a new food order from the customer. The customer personnel will be prompt for information that is needed which is the customer's name, address, contact number, and what food the customer had ordered.

Users are able to view the status of the order which is assigned with the letter "N" by default. The system will assign an order number for every order that is created. All of the orders can be viewed under view order information function and change the status of the order. The system will repeat until the user selects to exit from the system.

The following documentation will be the explanations of the program which are the data structure concept implemented in this program and the algorithm of it. The input and output of the program will also be shown in the way of screenshots when it was executing

Explanation: Data structures and function implemented with code snippet

There is only one data structure that was applied in this program which is the linked list data structure that is used to store and manage the data of the program.

The linked list data structure that had implemented in this program is single linked list which is a dynamic data structure that implement with the pointers. A linked list is a sequence of links that hold items and each of the links holds a connection to some other link. Every data field of the linked list contains is a space for storing the data and the reference field is a space for the pointer that link to the next member of the list. The advantage of using a linked list compare with arrays is there is no need to define the initial size of the variable and it is easier to use in the sense of the insertion and deletion of data. The array will be less efficient when the data size became bigger and it takes more time to insert or delete data. However, an array is suitable for data size that is smaller.

First, we are going to create a header file that will store all the function that will be using in the main cpp file. A structure is created that contains the variable that we are going to use in the code.

```
using namespace std;
struct orderinfo
{
    int orderNumber, itemCode,itembelongto;
    string itype;
    int quantity;
    float costPerItem;
    float totCost=0.00;
    char fulfillOrder;
    string recipient_name;
    string address;
    int telNum;

    orderinfo* next;
    char selection;
};
```

Next, we create another structure that save function will be used later. In this case, we are creating three different pointers that point to a different head also size variable created differently to determine the size of the list.

```
struct linkedList
{
    orderinfo* head;
    orderinfo* itemhead;
    orderinfo* fullhead;
    int fullsize;
    int itemsize;
    int size;
```

The value of the variable is initialized in the constructor of the structure.

```
linkedList()
{
    this->size = 0;
    this->head = nullptr;
    this->itemsize = 0;
    this->itemhead = nullptr;
    this->fullsize = 0;
    this->fullhead = nullptr;
}
```

We create a function that adds a new order to the list. This function will take three parameters and pass on the value to the variable. This function will take a new node that is used to store the data and the pointer that points to the head variable. This will make the next item be added at the beginning of the list.

```
void newOrder(string name, string addr, int telno)
{
    orderinfo* norder = new orderinfo;
    norder->orderNumber = size + 1;
    norder->fulfillOrder = 'N';
    norder->recipient_name = name;
    norder->address = addr;
    norder->telNum = telno;

    norder->next =head;
    head = norder;
    size++;
}
```

The order number is equal to the list size plus one and order status will be assigned character 'N' by default. The order number makes the other function works easier and will be explained in the document.

The other function is adding item purchases to the list. It works the same with the newOrder function the difference between it is this function takes five parameters.

```
void newItem(int icode, string itype, int quan, float price,int ib)
{
    orderinfo* newItem = new orderinfo;
    newItem->itembelongto = ib;
    newItem->itemCode = icode;
    newItem->itype = itype;
    newItem->quantity = quan;
    newItem->costPerItem = price;
    newItem->next =itemhead;

    itemhead = newItem;
    itemsize++;
}
```

```
void displayrecipient(string a)
{
   orderinfo* current = head;
   while (current != NULL)
    {
      if (current->recipient_name==a)
      {
         cout << "Order Number :" << current->orderNumber << endl;
         cout << "Order Status :" << current->fulfillOrder << endl;
         cout << "Customer status " << endl;
         cout << "Customer Name :" << current->recipient_name << endl;
         cout << "Customer address :" << current->address << endl;
         cout << "Customer Telephone number: " << current->telNum << endl;
         current = current->next;
      }
      else
      {
         cout << "No record found in the list" << endl;
      }
   }
}
```

This displayrecipient function will display the detail of the recipient customer. First, we need to declare a pointer variable and point it to the head address which s null. Then it will loop if the pointer variable not equal to null. We pass a string parameter to the function which is used to locate which customer the function is looking for. Inside the while loop if the current->recipent_name is equal to the parameter it will print out only the information that is related to the customer.

There are other functions with the same name but different parameters. This is an overloading concept that avoids function with the same name. This function takes an integer as a parameter to perform the search function

```
void displayrecipient(int a)
{
    orderinfo* current = head;
    while (current != NULL)
    {
        if (current->orderNumber == a)
        {
            cout << "Order Number :" << current->orderNumber << endl;
            cout << "Order Status :" << current->fulfillOrder << endl;
            cout << "Customer status " << endl;
            cout << "Customer Name :" << current->recipient_name << endl;
            cout << "Customer address :" << current->address << endl;
            cout << "Customer Telephone number: " << current->telNum << endl;
            current = current->next;
        }
        else
        {
            cout << "No record found in the list" << endl;
        }
    }
}

void modifyrecipient(string name,string address,int tel,int a)
{
    orderinfo* current = head;
    while (current != NULL)
    {
        if (current->orderNumber == a)
        {
            current->recipient_name = name;
            current->address = address;
            current->telNum = tel;
            cout << "Customer Name :" << current->recipient_name << endl;
            cout << "Customer address :" << current->address << endl;
            cout << "Customer Telephone number: " << current->telNum << endl;
            current = current->next;
        }
        else
        {
            cout << "No record found in the list" << endl;
        }
    }
}
```

The function above is used to modify the data. Customer information that needed to be changed is passed as parameters. When the current variable order number is founded it will update change the information to the data that the user wishes to modify.

```
void deleteorderat(string a)
{
   orderinfo* current = head;
   orderinfo* prev = head;

   bool found = false;
   while (current != NULL)
   {
      if (current->recipient_name == a)
      {
         found = true;
         break;
      }
      else
      {
         prev = current;
         current = current->next;
      }
   }
   if (found)
   {
      if (current == head)
      {
         head = head->next;
         deleteitembelongto(current->orderNumber);
         delete current;
      }
      else
      {
         prev->next = current->next;
         deleteitembelongto(current->orderNumber);
         delete current;
      }
   }
   else
   {
      cout << " Please check the spelling" << endl;
   }
```

This function will delete the order when the user wish to. It consists of two-part, the first part will look for the order that the user wish to delete and the second part will delete the record. Inside this function will declare two pointers variable which is current and prev and one locale variable “found” which is false by default. The while loop was used the look over the whole list and will set the “found” variable to true if it gets the result. If it is true it will delete the current data and delete the item belongs to it by using “deleteitembelongto” function.

```
void displayorder_item()
{
    orderinfo* current = head;
    orderinfo* itemcurrent = itemhead;
    if (current == NULL)
    {
        cout << "No record in the list" << endl;
    }
    else
    {
        while (current != NULL)
        {
            cout << "Order Number :" << current->orderNumber << endl;
            cout << "Order Status :" << current->fulfillOrder << endl;
            cout << "Customer status " << endl;
            cout << "Customer Name :" << current->recipient_name << endl;
            cout << "Customer address :" << current->address << endl;
            cout << "Customer Telephone number: " << current->telNum << endl;

            float total = 0.0;
            float deliveryc=0.0;
            float totalpay=0.0;

            cout << "Item order" << endl;
            while (itemcurrent!=NULL)
            {
                if (current->orderNumber==itemcurrent->itembelongto)
                {
                    cout << itemcurrent->itemCode << "\t" << itemcurrent->itype << "\t" << itemcurrent->quantity << "\t" << itemcurrent->costPerItem << endl;
                    total=total+ itemcurrent->costPerItem;
                    deliveryc = total * 0.05;
                    totalpay = total * 1.05;
                    itemcurrent = itemcurrent->next;
                }
                else
                {
                    break;
                }

            }
            cout << "Total amount :" << total << endl;
            cout << "Delivery Charge : " << deliveryc << endl;
            cout << "Total Pay : " << totalpay << endl << endl;
            current = current->next;

        }
    }
}
```

The function above will print out all the data that are available inside the list including the item customer bought. In this situation, every item has its owns item belongs to ID and it is easy to make a classification based on the id. In the outer loop, it will go through the list and print out the data in it in the inner loop is for the item list. It will print out the element that belongs to the order number and calculate the total price and also the delivery charge. The outer loop will execute until the current pointer points to NULL.

```
void fufilordermove()
{
    orderinfo* current = head;
    orderinfo* fullcurrent =fullhead;
    orderinfo* newfull = new orderinfo;
    newfull->orderNumber = current->orderNumber;
    newfull->fulfillOrder = 'F';
    newfull->recipient_name = current->recipient_name;
    newfull->address = current->address;
    newfull->telNum = current->telNum;
    deleteorderat(current->orderNumber);
    newfull->next = fullhead;
    fullhead = newfull;
    fullsize++;
}
```

This function copies the data from the order list to a new list that store the order that has been fulfilled and delete the order from the order list. The function that used to delete the function is the “deleteorderat” function.

```
void displayfullfillorder()
{
    orderinfo* fullcurrent = fullhead;
    if (fullcurrent==NULL)
    {
        cout << "No order fullfilled" << endl;
    }
    else
    {
        while (fullcurrent!=NULL)
        {
            cout << "Order Number :" << fullcurrent->orderNumber << endl;
            cout << "Order Status :" << fullcurrent->fulfillOrder << endl;
            cout << "Customer status " << endl;
            cout << "Customer Name :" << fullcurrent->recipient_name << endl;
            cout << "Customer address :" << fullcurrent->address << endl;
            cout << "Customer Telephone number: " << fullcurrent->telNum << endl;
            fullcurrent = fullcurrent->next;
        }
    }
}
```

This function display the order that had been fulfilled. It is using the same way as the other display function.

```
bool search()
{
    orderinfo* current = head;

    if (current == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

.

The search function is a boolean type function that returns true or false value when the condition meets. It is used to validate the input of the user and check if the list is having any data or did not have any record in the list.

```
void invalidinput()
{
        cin.clear();
        cin.ignore(1000, '\n');
        cout << "Invalid input, try again" << endl << endl;
}
```

The invalidinput function is used to validate the input made by the user. cin.clear() will clear the error on cin and cin.ignore() will skip to the next line and ignore the current line.

```
        FOOD ORDER AND DELIVERY SYSTEM
Pick one of the service below:
1.      Create a new order
2.      Search for an order
3.      View orders information
4.      View delivery list
5.      Quit

Your selection:
```

This will prompt the user to pick one of the service.

Select the first service which is to create new order and will show up the information that needs to be filled.

```
Insert Customer Name: Mark
Insert Customer Address: lavillas
Insert Customer telephone number: 123456789
```

Then user can order the food.

```
        [Item Menu]
1.      Burger                  $6.50 per one
2.      Spagethetti             $7.50 per one
3.      Coke                    $1.50 per one
4.      Ice-cream               $1.00 per one
5.      Apple Pie               $2.80 per one
6.      Drinking water          $2.00 per one
7.      Americano Espresso      $6.50 per one
8.      Cash Total
Insert item code: 1
Insert item quantity: 4
Insert item code:
```

Insert the item code and also the quantity of it. Insert number 8 to finish the order and will go back to the home page.

Next user can search for order by using order number or customer name.

```
        FOOD ORDER AND DELIVERY SYSTEM
Pick one of the service below:
1.      Create a new order
2.      Search for an order
3.      View orders information
4.      View delivery list
5.      Quit

Your selection:2

Search order number or recipient name in order to proceed to next step:
1.Name  2.Order Number
Selection:
```

After that will show up the action the user can take. Press 1 to delete the record or modify the record. User can stop the service by pressing 3 back to home page.

```
        FOOD ORDER AND DELIVERY SYSTEM
Pick one of the service below:
1.      Delete an order
2.      Modify for an order
3.      Back <<

Your selection:
```

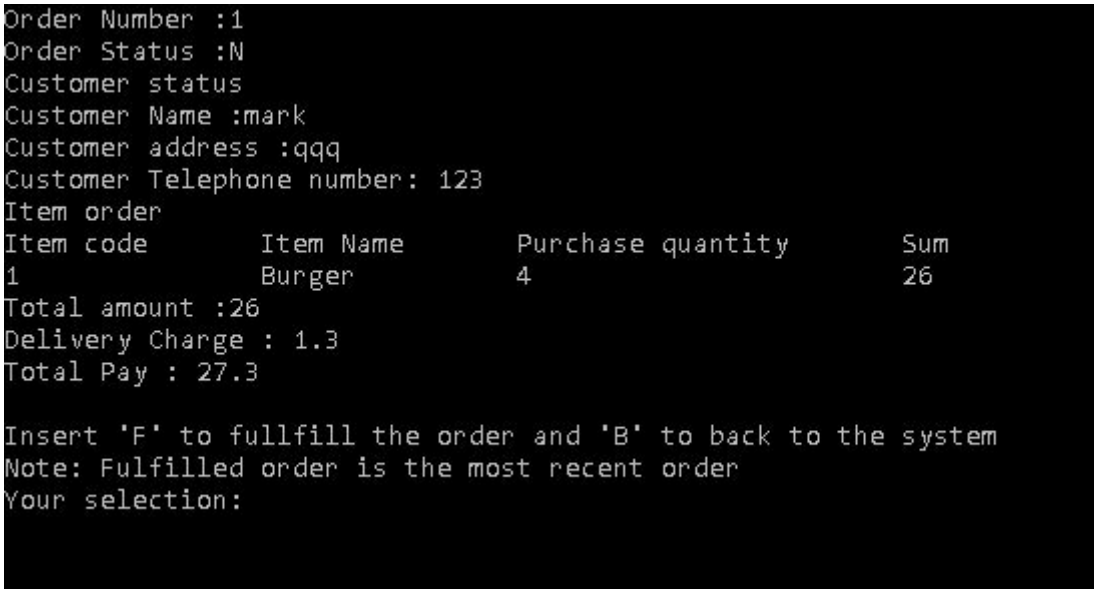
If choosing delete an order it will show delete success message and back to homepage.

```
Your selection:
1
Delete Success
Delete Success
```

If choose to modify the order it will show what information can be modified and user have to insert the new information. It will prompt the user to confirm the modification select n if the user regret.After complete the action it will show modify success and back to the homepage..

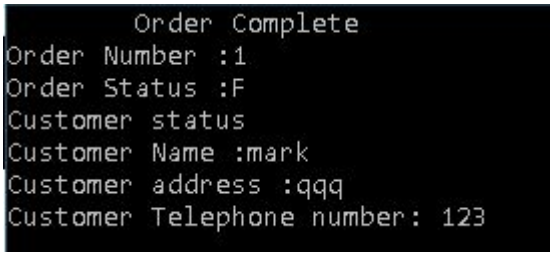
```
This is what you can modify:
Original :
Order Number :1
Order Status :N
Customer status
Customer Name :Mark
Customer address :lavillas
Customer Telephone number: 123456789

Name : mark
Customer new address :qqq
Customer new telephone number :123
Type 'Y' for confirmation and 'N' for cancel service
```

User can view the current record that available by choosing 3. It will show all the orders that available including customer information and ordered item including the total price and delivery charge.



User can choose to fulfill the order or back to the homepage. If user choose to fulfill the order it will be removed from the list and moved the fullfill list.

In order to view the fulfilled order press number 4 on the home page.



The order status is changed to F which mean the order is complete.

Source Code

```
int main()
{
	linkedList orderlist;
	orderinfo* oinfo = new orderinfo;
	string name;
	string addr;
	int telno;
	int icode;
	string itype;
	string searchname;
	int quan;
	float price;
	int selectfunc,secondselect;
	int selectitem;
	int itembelong;
	int searchinfo, searchnum;
	do
	{
		displaysystem();

		if(cin >> selectfunc)
		{
			cout << endl;
			switch (selectfunc)
			{
			case 1:
				system("CLS");
				cout << "Insert Customer Name: ";
				cin >> name;
				cout << "Insert Customer Address: ";
				cin >> addr;
				cout << "Insert Customer telephone number: ";
				if (cin >> telno)
				{
					system("CLS");
					displaymenu();
					orderlist.newOrder(name, addr, telno);
					itembelong = orderlist.size;
					do
					{
						cout << "Insert item code: ";
						if (cin >> selectitem)
						{
							switch (selectitem)
							{
```

```
                                        case 1:
                                                icode = 1;
                                                itype = "Burger";
                                                cout << "Insert item quantity:
";
                                                cin >> quan;
                                                price = 6.50 * quan;
                                                orderlist.newItem(icode, itype,
quan, price, itembelong);
                                                break;
                                        case 2:
                                                icode = 2;
                                                itype = "Spagethetti";
                                                cout << "Insert item quantity:
";
                                                cin >> quan;
                                                price = 7.50 * quan;
                                                orderlist.newItem(icode, itype,
quan, price, itembelong);
                                                break;
                                        case 3:
                                                icode = 3;
                                                itype = "Coke";
                                                cout << "Insert item quantity:
";
                                                cin >> quan;
                                                price = 1.50 * quan;
                                                orderlist.newItem(icode, itype,
quan, price, itembelong);
                                                break;
                                        case 4:
                                                icode = 4;
                                                itype = "Ice-cream";
                                                cout << "Insert item quantity:
";
                                                cin >> quan;
                                                price = 1.00 * quan;
                                                orderlist.newItem(icode, itype,
quan, price, itembelong);
                                                break;
                                        case 5:
                                                icode = 5;
                                                itype = "Apple Pie";
                                                cout << "Insert item quantity:
";
                                                cin >> quan;
                                                price = 2.80 * quan;
                                                orderlist.newItem(icode, itype,
```

```
quan, price, itembelong);

                                                        break;
                                                case 6:
                                                        icode = 6;
                                                        itype = "Drinking water";
                                                        cout << "Insert item quantity:
";
                                                        cin >> quan;
                                                        price = 2.00 * quan;
                                                        orderlist.newItem(icode, itype,
quan, price, itembelong);
                                                        break;
                                                case 7:
                                                        icode = 7;
                                                        itype = "Americano Espresso";
                                                        cout << "Insert item quantity:
";
                                                        cin >> quan;
                                                        price = 6.50 * quan;
                                                        orderlist.newItem(icode, itype,
quan, price, itembelong);
                                                        break;
                                                case 8:
                                                        system("CLS");
                                                        break;
                                                default:
                                                        break;
                                                }

                                        }
                                        else
                                        {
                                                invalidinput();
                                                system("CLS");
                                        }

                                } while (selectitem != 8);
                                itembelong++;
                        }
                        else
                        {
                                invalidinput();
                        }

                        break;
                case 2:
                        char yn;
```

```
                                cout << "Search order number or recipient name in order to
proceed to next step: " << endl;
                                cout << "1.Name\t2.Order Number" << endl;
                                cout << "Selection: ";
                                cin >> searchinfo;
                                switch (searchinfo)
                                {

                                case 1:
                                        cout << "Search:";
                                        cin >> searchname;
                                        searchfuncdisplay();
                                        cout << endl;
                                        if (cin>>secondselect)
                                        {
                                                switch (secondselect)
                                                {
                                                case 1:
                                                        orderlist.deleteorderat(searchname);
                                                        cout << "Delete Success" << endl;
                                                        break;
                                                case 2:
                                                        system("CLS");
                                                        cout << "This is what you can modify:
" << endl;
                                                        cout << "Original :" << endl;
                                                        orderlist.displayrecipient(searchname);
                                                        cout << endl;
                                                        cout << "New information" << endl;
                                                        cout << "Name : ";
                                                        cin >> name;
                                                        cout << "Customer new address :";
                                                        cin >> addr;
                                                        cout << "Customer new telephone
number :";
                                                        cin >> telno;
                                                        cout << "Type 'Y' for confirmation and
'N' for cancel service" << endl;
                                                        cin >> yn;
                                                        if (yn == 'y' || yn == 'Y')
                                                        {

orderlist.modifyrecipient(name, addr, telno, searchname);
                                                                cout << "Modify success" <<
endl;
                                                                break;
                                                        }
                                                        else if (yn == 'n' || yn == 'N')
```

```
                                        {
                                                break;
                                        }
                                        else
                                        {
                                                invalidinput();
                                        }

                                default:
                                        break;
                                }
                                break;
                        }

                        else
                        {
                                invalidinput();
                        }
                case 2:
                        cout << "Search:";
                        cin >> searchnum;
                        searchfuncdisplay();
                        cout << endl;
                        if (cin >> secondselect)
                        {
                                switch (secondselect)
                                {
                                case 1:
                                        orderlist.deleteorderat(searchnum);
                                        cout << "Delete Success" << endl;
                                        break;
                                case 2:
                                        system("CLS");
                                        cout << "This is what you can modify:
" << endl;
                                        cout << "Original :" << endl;
                                        orderlist.displayrecipient(searchnum);
                                        cout << endl;
                                        cout << "Name : ";
                                        cin >> name;
                                        cout << "Customer new address :";
                                        cin >> addr;
                                        cout << "Customer new telephone
number :";
                                        cin >> telno;
                                        cout << "Type 'Y' for confirmation and
'N' for cancel service" << endl;
                                        cin >> yn;
```

```
                                        if (yn == 'y' || yn == 'Y')
                                        {

orderlist.modifyrecipientint(name, addr, telno, searchnum);
                                                cout << "Modify success" <<
endl;
                                        }
                                        else if (yn == 'n' || yn == 'N')
                                        {
                                                break;
                                        }
                                        else
                                        {
                                                invalidinput();
                                        }

                                default:
                                        break;
                                }
                                break;
                        }
                        else
                        {
                                invalidinput();
                        }
                case 3:
                        break;
                default:
                        break;
                }
                break;

        case 3:
                system("CLS");
                orderlist.displayorder_item();
                char fullfill;

                if (orderlist.search())
                {
                        break;
                }
                else
                {
                        cout << "Insert 'F' to fullfill the order and 'B' to back
to the system" << endl;
                        cout << "Note: Fulfilled order is the most recent
order" << endl;
                        cout << "Your selection: ";
```

```
                                        if (cin >> fullfill)
                                        {
                                                if (fullfill == 'F' || fullfill == 'f')
                                                {
                                                        orderlist.fufilordermove();
                                                        cout << "Fullfill order success" <<
endl << endl;
                                                        system("CLS");
                                                }
                                                else
                                                {
                                                        system("CLS");
                                                        break;

                                                }
                                        }
                                        else
                                        {
                                                invalidinput();
                                        }
                                }
                                break;
                        case 4:
                                system("CLS");
                                cout << "\tOrder Complete"<<endl;
                                orderlist.displayfullfillorder();
                                break;
                        case 5:
                                system("CLS");
                                cout << "Thank you for using" << endl;
                                exit(0);
                        default:
                                break;
                        }
                }
                else
                {
                        invalidinput();
                        system("CLS");
                }

        } while (selectfunc!=5);
}
```