**struct node**

declare itemNum;

declare itemName;

declare itemCost;

declare itemQty;

declare struct product next pointer, initialize to NULL

declare struct product previous pointer, initialize to NULL

**Function main**

***Algorithm***

1. Create a LinkedList object
2. Create an Inventory object
3. Call BuildProductList()
4. Display menu: 1. Admin, 2. Customer, 3. Exit
5. Use switch statement
6. Case 1: admin

Create an admin object

Call admin.validateUsernameandPassword

If admin is validated

Display "You are currently in admin mode!"

Do

Display “to add product enter add, to delete product enter delete, to update product qty enter update”

Store output into adminOption

Call adminOptions()

Display “exit admin mode enter exit”

Store output into endAdmin

While(endAdmin != “exit”)

Call updateInventory() function

1. Case 2: customer

Create a customer object

Call readFile()

Display “If you would like to continue to checkout, enter 0.”

Display “If you would like to view your cart, enter cart.”

For (count=0, count < lengthList , count++)

Set productNum to productPurchase()

Validate productNum

If (enterCart=true)

customer.displayCart

If (productNum=0)

customer.getInfo

customer.printInvoice

deconstructor deletes linked list

Break loop

Else

Set productExists to searchProduct()

If (productExists=false)

Display “product does not exist”

Else

Do

Display “What is the qty wanting to be purchased”

Store output into productQty

Validate productQty

While(qtyExists=false)

If(qtyExists=true)

Display “product has been added to your shopping cart”

If product has already been purchased

Customer.updateProductQty

Customer.addItemCart

1. Case 3:Exit

Exits switch

Class function to update the ProductData.csv

**Function ReadFile:** reads file from productData.csv and builds a linked list from the file

***Parameters:*** inFile by reference

***Processing:*** tempProduct, titemNum, titemName, titemCost, titemQty, titemNum1, titemName1, titemQty1, titemCost1

***Output:*** N/A

***Return type:*** void

***Algorithm***

1. Open inFile
2. If inFile is unable to open

End function

1. If file is able to open
2. Create a linked list object
3. Create an inventory object

While (end of File has not been reached)

Use getLine to get the itemNum

Inventory.getItemNum

Use getLine to get the itemProduct

Inventory.getItemName

Use getLine to get the itemCost

Inventory.getItemCost

Use getLine to get the itemQty

Inventory.getItemQty

LinkedList.addNode(int, string, double, int)

1. Close file

**Function productPurchase:** asks the user which product they would like to purchase and return the product number

***Parameters****:* string cart by reference

***Processing:*** productNumber, answer

***Output:*** productNumber

***Return type:*** int

**Algorithm**

1. Display “What product would you like to purchase”

Store output in answer

1. If (answer=”cart”)

Store cart as true

Else

Store cart as false

Convert answer to an int and store it in productNumber

Return productNumber

**Function adminOptions:** if the admin enters admin status, they are given the option to add product, delete product, update productsQty

***Parameters:*** Admin admin

**Processing:** nitemNum, nitemName, nitemQty, nitemCost

***Output:*** n/a

***Return type:*** void

***Algorithm***

1. If (adminOption = “add”)

Ask the user for the new product number, cost of the new product, and qty of new product

Store output in nitemNum, nitemName, nitemQty

admin.addProduct

1. If(adminOption = “delete”)

Ask the user “what is the product number you would like to delete”

Store ouput in ditemNum

If (searchProduct()=true)

admin.deleteProduct

Else

Display “the product does not exist”

1. If(adminOption = “update”)

Ask the user “what is the product number the would like to update”

Store ouput in nitemNum

If (searchProduct()=true)

Ask the user “new qty of the product”

admin.insertQty

Else

Display “the product does not exist”

|  |
| --- |
| **Linked List** |
| - head: node\* |
| + LinkedList():  + LinkedList(newNode: node\*):  + ~LinkedList():  + addNode(num:int, name:string, cost:double, qty:int):void  + deleteNode():void  + operator+=(list: LinkedList&): LinkedList |

|  |
| --- |
| **Inventory** |
| - itemNum: int  - itemName: string  - itemCost: double  - itemQty: int |
| + Inventory():  + Inventory(newNode: node\*):  + ~Inventory():  + getItemNum():int const  + getItemName():string const  + getItemCost():double const  + getItemQty():int const  + setItemQty(productNum: int, nItemC ):void  + setItemCost(productNum: int, nItemQty: double):void  + updateInventory(list: LinkedList):void |

|  |
| --- |
| **Customer** |
| - firstName:string  - lastName:string  - address:string  - city:string  - state:string  - zipCode:int  - invoiceFile: string |
| + Customer():  + Customer(newNode: node\*):  + ~Customer():  + Shopping Cart (LinkedList):void  + displayCart():void const  + addItemCart(item: int):void  + deleteItemCart(item: int):void  + updateItemQty(newQty:int):void  + getInfo():void  + printInvoice(invoiceFile:string):void |

|  |
| --- |
| **Admin** |
| - username:string  - password:string |
| + Admin():  + ~Admin():  + addProduct(num: int, cost: double, qty: int):void  + deleteProduct(productNum: int):void  + insertQty(qty: int):void  + validateUsernamePassword(username:string, password:string): bool |