**Documentation**

**Ahmed Bin Khalid**

Table of Contents

1 – Introduction

2 – Algorithms

* Insertion function
* Search function
* Remove function
* Display function

3 – Explanation of the code

* Main methods
* Insertion Function
* Display Function
* Remove Function
* Search Function
* Search by Category Function

4 – Pseudocode

5 – Sample Input and Output

**Introduction:**

The concept was to create a program which would have to make an inventory system that uses a linked list. The program should be able to keep track of the SKU, product name, product price, quantity and category of the product.

We should be able to Insert, Display, Delete and Search. These are all basic functions that a linked list offers.

**Algorithms:**

**Insertion function**

Begin

Create new nodes

Add the data (prodname, quantity etc) into new nodes

If statement

When no nodes available in the list

The new node is equal to the first node

Else statement

Find a node which has a value that is greater or equal to the new value or at the end of the list. Insert the new node before the node you found or insert the new node at the end of the list if no node was found.

**Search Function**

Begin

Declare nodeptr and previous node

Assign the nodeptr (nodepointer) to list head

While loop when node point’s SKU is not equal to input

The node pointer is moved to the next node

End while

Display the product name, product price, quantity and category of the node pointed to by the node pointer

**Search by Category Function**

Begin

Declare nodeptr

Assign the nodeptr (nodepointer) to list head

While loop when node pointer is not equal to null

If statement to make nodePtr point to sku equal to findSKU

End while

**Remove Function**

Begin

Declare nodeptr

Assign the nodeptr (nodepointer) to list head

If head matches with the argument

nodePtr points to the head->next, if it matches then delete head and replace it with nodePtr

While nodePtr is not null and nodePtr -> SKU is not equal to sk2

Then the previousNode will point to nodePtr

NodePtr will be equal to nodePtr link to the next value

End while

The previousNode -> next is assigned to the nodePtr -> next, then delete the nodePtr

**Display Function**

Begin

Declare nodeptr

Make nodePtr equal to head

While nodePtr is not NULL

The category inside the list must be equal to the input

So display SKU and Product name

NodePtr will travel to the next node

End While

**Explanation of the code**

**Main Methods**

struct listnode

{ //DECLARATIONS

string SKU, prodName, cat;

double price;

int quantity;

struct listnode \*next;

};

listnode \*head;

This code is necessary to create a nodelist, the elements that are needed (string, float, int) are also declared through it.

void main()

int amount, quantity, choice; //DECLARE ITEMS

string SKU, prodName, cat, findSKU, removeSKU, searchCategory;

double price;

char ans = 'Y';

This is the main() method that is used to declare the int, string and float values.

insertNode("ABC01", "PARTY DRESS", 3.50, 60, "Female");

insertNode("ABC02", "BUSINESS SUIT", 2.00, 100, "Male");

Dummy entries only for demonstration purposes

while(ans=='Y'||ans=='y') //WHILE LOOP FOR INPUT

{

cout << "MENU:" << endl; //MENU WITH OUTPUT

cout << "PRESS 1 TO INSERT SKU" << endl;

cout << "PRESS 2 TO DELETE SKU" << endl;

cout << "PRESS 3 TO SEARCH SKU" << endl;

cout << "PRESS 4 TO SEARCH CATEGORY" << endl;

cout << "PRESS 5 TO SHOW ALL ITEMS" << endl;

cin >> choice;

This code will display a main menu of sorts after each time a function is run, it will ask for number from 1-5 from this the user can choose which ever option they desire.

case 1: cout << "ENTER AMOUNT OF ITEMS" << endl;

cin >> amount;

for (int i = 0; i < amount; i++)

Print text to request user to input total no of items, there is a for loop that will run using the amount that is entered

switch (choice) //SWITCH STATEMENT

{

case 1: cout << "ENTER AMOUNT OF ITEMS" << endl;

cin >> amount;

for (int i = 0; i < amount; i++)

{

cout << "ENTER SKU NUMBER:" << endl;

getchar();

cin >> SKU;

cout << "ENTER PRODUCT NAME:";

getchar();

cin >> prodName;

cout << "ENTER PRODUCT PRICE:";

cin >> price;

cout << "ENTER QUANTITY:";

cin >> quantity;

cout << "ENTER CATEGORY [MALE OR FEMALE]:";

getchar();

cin >> cat;

insertNode(SKU, prodName, price, quantity, cat);

} break;

case 2: cout << "REMOVE SKU:" << endl;

cin >> removeSKU;

deleteNode(removeSKU); break;

case 3: cout << "SEARCH SKU:" << endl;

cin >> findSKU;

showSKU(findSKU); break;

case 4: cout << "SEARCH WITH CATEGORY [MALE OR FEMALE]:" << endl;

cin >> searchCategory;

showCat(searchCategory); break;

case 5: showAll(); break;

default: cout << "ERROR OCCURED"; break;

}

cout << "\nWOULD YOU LIKE TO CONTINUE?(Y or N)";

cin >> ans;}

After choosing an option the switch case statement will call that particular function.

**Insertion Function**

void insertNode(string sku, string ITEMNAME, double PRICE, int QUANTITY, string CATEGORY) {

listnode \*newNode, \*nodePtr = NULL, \*previousNode = NULL;

First and formost we have to create a node to hold all the new values.

newNode = new listnode;

newNode->SKU = sku;

newNode->prodName = ITEMNAME;

newNode->price = PRICE;

newNode->quantity = QUANTITY;

newNode->cat = CATEGORY;

Then we store their respective data into new nodes.

if (!head) //IF NOT EQUAL TO HEAD THEN

{

head = newNode; //INITIALIZE NODEPTR TO HEAD

newNode->next = NULL;

}

Or we can insert a new node by initializing the nodePtr to the head of the list.

while (nodePtr != NULL && nodePtr->SKU < sku)

{

previousNode = nodePtr;

nodePtr = nodePtr->next;

This code will case all the nodes which have a sku value that is less than sku to skip. If the while condition is true, then previous node will be equal to node pointer and the node pointer will be equal to the nodePtr->next

if (previousNode == NULL) //IF PREVIOUS NODE IS NULL

{

head = newNode; //HEAD IS EQUAL TO NEW NODE

newNode->next = nodePtr;//MOVE TO NEXT NODE

}

else

{

previousNode->next = newNode;

newNode->next = nodePtr;

}

If a new node is meant to be the first in the list then insert it before all other nodes. Compare the previousNode with NULL. If its true than head is equal to newNode and newNode->next is equal to nodePtr. Otherwise the previousNode-> is equal to newNode and the newNode is equal to nodePtr

**Display function**

void showAll() { //METHOD CALLED BY FUNCTION IN SWITCH CASE

listnode \*nodePtr;

nodePtr = head; //ASSIGN NODEPTR TO HEAD

while (nodePtr != NULL) { //WHILE NODEPTR IS NOT NULL

It will test the node pointer with the while loops conditional statement that nodePtr is not equal to NULL

cout << nodePtr->SKU << endl;

cout << "NAME: " << nodePtr->prodName << endl;

cout << "PRICE: " << nodePtr->price << endl;

cout << "QUANTITY: " << nodePtr->quantity << endl;

cout <<"CATEGORY: " << nodePtr->cat << "\n" << endl;

nodePtr = nodePtr->next;

This segment will display the data from the list. It will first declare the showAll as the function that traverses the list, as it moves it will be displaying the prodname, price, quantity and category member of each node.

**Remove function**

void deleteNode(string removeSKU) //FUNCTION WITH STRING AS AN ARGUMENT

{

bool error\_check = 0;

listnode \*nodePtr, \*previousNode;

The link nodelist is linked to the nodePr and previousNode

if (head->SKU == sk2){ //check with the head of linked list to see if it matches with the argument

nodePtr = head->next;

delete head;// if it matches then delete it

head = nodePtr; //replace head with nodeptr

} //else will run id the head doesnt match with the argument

If sku to sk2 is equal then the statement is true and the nodePtr will link to the next value and the head will be deleted making nodePtr the new head.

else

{

nodePtr = head;

previousNode = NULL;

// travel up-to the node which has a data as a string, passed as argument

while (nodePtr != NULL && nodePtr->SKU != sk2)

{

previousNode = nodePtr;

nodePtr = nodePtr->next;

} previousNode->next = nodePtr->next;

delete nodePtr;// if found then delete that node

}

}

If thenodePtr is not Null and the value of SKU is not equal to sk2, the while statement will be run where the previousNode will replace the current value of the nodePtr, and the nodePtr will now link to the next value. Since the while statement is completed, the current value of nodePtr will be deleted and replaced with the new value.

**Search function**

void showCat(string searchcategory)

{

listnode \*nodePtr, \*previousnode;

nodePtr = head;

bool error\_check = 0;

while (nodePtr != NULL)

{

if (nodePtr->cat == searchcategory)

{

cout << nodePtr->SKU << endl;

cout << "CATEGORY: " << nodePtr->cat << endl;

cout << "PRODUCT NAME: " << nodePtr->prodName << endl;

error\_check = 1;

}

nodePtr = nodePtr->next;

if (nodePtr != NULL && error\_check != 1)

{

cout << "NO CATEGORY FOUND" << endl;

}

If the NodePtr is not equal to null, the pointer will traverse all nodes in the list. An if statement is used to check that the category same as the user input, int the case that it is the same, it will display the sku and its product name, then it will run to the next one. It will loop continually until nodePtr is null.

**Search by category function**

void showSKU(string findSKU) {

listnode \*nodePtr;

nodePtr = head;

while (nodePtr != NULL) {

if (nodePtr->SKU == findSKU)

{

While the node pointer is not equal to null the if statement is run where SKU is equal to findSKU.

cout << "PRODUCT NAME: " << nodePtr->prodName << endl;

cout << "UNIT PRICE: " << nodePtr->price << endl;

cout << "QUANTITY: " << nodePtr->quantity << endl;

cout << "CATEGORY: " << nodePtr->cat << "\n\n";

}

Hence the user is shown the appropriate values and their titles

nodePtr = nodePtr->next;

}

if (nodePtr == NULL && error\_check != 1)

{

cout << "ERROR OCCURED" << endl;

}

Otherwise this statement till be run which means that nodePointer is equal to null and error check is not 1 so it will display ERROR OCCURRED.

**Pseudocode**

STRUCT listnode

Declare SKU as string

Declare itemName as string

Declare price as double

Declare quantity as int

Declare category as string

STRUCT listnode \*next

Listnode \*head

METHOD showAll()  
{

listnode \*nodePtr;

declare nodePtr as head

WHILE nodePtr is not NULL (empty)

display nodePtr points to SKU

display “NAME: “

display nodePtr points to prodName

display “PRICE: “

display nodePtr points to price

display “QUANTITY: “

display nodePtr points to quantity

display “CATEGORY:”

display nodePtr points to cat

nodePtr = nodePtr points to next nodePtr (traverse)

END WHILE

}

METHOD showSku(string searchsku)

{

listnode \*nodePtr

declare error\_check as 0

declare nodePtr as head

WHILE nodePtr is not NULL (empty)

If nodePtr Points to SKU is equal searchsku

display “PRODUCT NAME:”

display nodePtr Points to itemName

display “UNIT PRICE: “

display nodePtr Points to price

display “QUANTITY:”

display nodePtr Points to quantity

display “CATEGORY:”

display nodePtr Points to category

set error\_check as 1

END IF

nodePtr = nodePtr points to next nodePtr (traverse)

END WHILE

IF nodePtr is equal to NULL and error\_check is not equal 1

display “No SKU found”

END IF

}

METHOD showCat(string searchcategory)

{

listnode \*nodePtr

listnode \*previousnode

declare error\_check as 0

declare nodePtr as head

WHILE nodePtr is not NULL (empty)

If nodePtr Points to category is equal searchcategory

Display nodePtr Points to SKU

display “PRODUCT NAME:”

display nodePtr Points to prodName;

display “CATEGORY:”

display nodePtr Points to category

set error\_check as 1

END IF

nodePtr = nodePtr points to next nodePtr (traverse)

END WHILE

IF nodePtr is equal to NULL and error\_check is not equal 1

display “No category found for the product.”

END IF

}

METHOD insertNode(string sku, string ITEMNAME, double PRICE, int QUANTITY, string CATEGORY))

{

listnode \*newNode

listnode \*nodePtr equal NULL

listnode \*previousNode equal NULL

newNode equal to new listnode;

declare newNode Points to SKU as sku

declare newNode Points to prodName as ITEMNAME

declare newNode Points to price as PRICE

declare newNode Points to quantity as QUANTITY

declare newNode Points to category as CATEGORY

IF not equal head

declare head as newNode

declare newNode Points to next as NULL

END IF

ELSE

declare nodePtr as head

WHILE nodePtr is not equal NULL and nodePtr Points to SKU is less than sku

declare previousNode as nodePtr

declare nodePtr as nodePtr Points to next

END WHILE

END ELSE

IF previousNode is equal to NULL

declare head as newNode

declare newnode Points to next as nodePtr

END IF

ELSE

declare previousNode Points to next as newNode

declare newNode Points to next as nodePtr

END ELSE

}

METHOD deleteNode(string removeSku)

{

listnode \*nodePtr

listnode \*previousNode

IF head Points to SKU is equal removeSku

declare nodePtr as head Points to next

delete head

declare head as nodePtr

END IF

ELSE

declare nodePtr as head

declare previousNode as NULL

WHILE nodePtr is not equal NULL and nodePtr Points to SKU is not equal deletesku

declare previousNode as nodePtr

declare nodePtr as nodePtr Points to next

END WHILE

declare previousNode Points to next as nodePtr Points to next

delete nodePtr

END ELSE

}

METHOD main(){

declare amount as int

declare quantity as int

declare choice as int

declare SKU as string

declare ProdName as string

declare category as string

declare searchSKU as string

declare removeSKU as string

declare searchCategory as string

declare price as double

declare ans as char and initialize to ‘Y’

insertNode("ABC01", "PARTY DRESS", 3.50, 60, "Female");

insertNode("ABC02", "BUSINESS SUIT", 2.00, 100, "Male");

WHILE ans is equal to ‘Y’ or ans is equal to ‘y’

display “MENU::”

display “PRESS 1 TO INSERT SKU.”

display “PRESS 2 TO DELETE SKU.”

display “PRESS 3 TO SEARCH SKU.”

display “PRESS 4 TO SEARCH CATEGORY.”

display “PRESS 5 TO SHOW ALL ITEMS”

CASE choice

1: display “Enter the amount of items”

enter amount

FOR int i equal 0; i less than amount; i++

display “ENTER SKU NUMBER.”

display “SKU(Example: SKU01):”

getchar()

enter SKU

display “ENTER PRODUCT NAME:

getchar()

enter itemName

display “ENTER PRODUCT PRICE:

getchar()

enter price

display “ENTER QUANTITY:"

getchar()

enter quantity

display “ENTER CATEGORY [MALE OR FEMALE]:

getchar()

enter category

insertNode(SKU, prodName, price, quantity, category)

ENDFOR

2: display “REMOVE SKU:”

enter removeSKU

deleteNode(removeSKU)

3: display “SEARCH SKU::”

enter findSKU

searchsku(findSKU)

4: display “SEARCH WITH CATEGORY [MALE OR FEMALE]:

enter searchCategory

showCat(searchCategory)

5: showAll()

default: display “ERROR OCCURED”

ENDCASE

display “WOULD YOU LIKE TO CONTINUE? (Y or N)”

enter ans

ENDWHILE

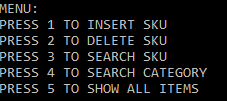
}

**Input and Output**

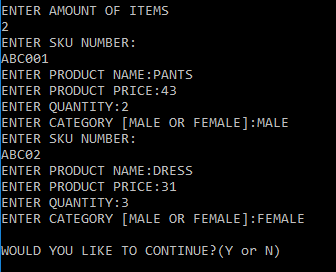
1) First we ask the user to enter the amount of items they want to store, in this instance we will have 2.

Screen%20Shot%202017-04-26%20at%2011.23.48%20AM.png

2) This menu will appear immediately to the user after they enter their value, it gives us some options that we can do by running their functions. Menu is based on switch case.



3) Now we will first press 1 to insert SKU, that will make the program prompt the user to enter first, the sku number, the product price, the quantity, the category and since we entered the amount of items as 2, the for loop will loop again asking the user for more data.

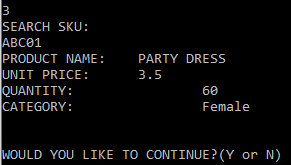


4) We ask the user finally if they would like to continue or not, if yes then it will bring back the menu so that the user can run more functions on their program.

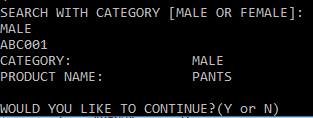
If “N” then it will end the program.

untitled%20folder%202/3.PNG

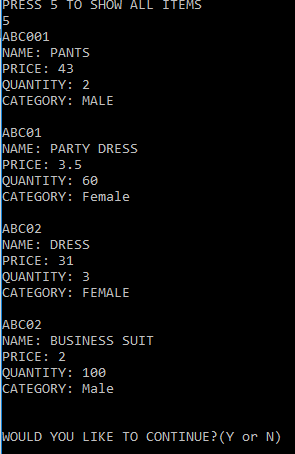
5) We will now run the third function which is to search sku. This will find the sku that we will input using nodes, it will travers the list looking for nodes that match the user input.



6) The fourth function will find all the values in any given category and display them, in this program we only had male and female products, but other categories can be added easily.



7) The showAll function is the fifth option in our program and will display all the items regardless of category or sku.



8) Now we press 2 and run the remove function, first it will ask for the sku to remove, user must input that. The function will run and the input would be removed immediately

untitled%20folder%202/REMOVE%207.PNG

9) As you can see from the below output, ABC01 or the SKU that we selected to remove is no longer there when you run the showAll function.

