**University of Michigan – Dearborn**

**CIS 200 – Computer Science 2**

**Project 03**

Nahrin Sharna

[nsharna@umich.edu](mailto:nsharna@umich.edu)

03/25/2019

# Source Code:

/\*

Author: Nahrin Sharna

Creation Date: 03/22/2019

Modification Date: 03/25/2019

Purpose: Insert, Search and delete operation on a circular doubly linked list

\*/

#include<iostream>

#include<string>

using namespace std;

struct node {

char letr;

node \* nextPtr,

\*prevPtr;

}\*head = NULL, \*r = NULL, \*tail = NULL;

void insert(char, node \*&);

void printListAscending(node \*);

void printListDescending(node \*);

void delete1(char);

bool search(char, node \*, node \*);

//main

int main() {

node \*p;

char item;

char choice;

char c,x;

int n;

node \*first = head;

//inserting list

cout << "Please enter the character or 0 to quit: ";

cin >> item;

while (item != '0') {

insert(item, head);

cout << "Please enter the character or 0 to quit: ";

cin >> item;

}

//printing the list

cout << "Your choice of printing(A or D): ";

cin >> choice;

if (choice == 'A') {

printListAscending(head);

}

else if (choice == 'D') {

printListDescending(head);

}

else {

cout << "Invalid option is chosen." << endl;

}

//delete any number

//delete function will call search funtion which will return false when the value does not exists.

cout << "Do you want to delete any value: ";

cin >> choice;

if (choice == 'Y') {

cout << "Enter the value you want to delete: ";

cin >> x;

delete1(x);

}

else if (choice == 'N') {

cout << "Thank you for using our software." << endl;

}

else {

cout << "Invalid option is chosen" << endl;

}

system("pause");

return 0;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/22/2019

Modification Date: 03/25/2019

Purpose: insert value in the list given by the user

\*/

void insert(char c, node \*&head) {

node \*temp = new node;

int count = 0;

node \*p;

p = head;

node \*first = head;

temp->letr = c;

temp->nextPtr = NULL;

temp->prevPtr = NULL;

if (head == NULL) {

head = temp;

temp->nextPtr = head;

temp->prevPtr = head;

p = head;

tail = temp;

}

else {

bool r1 = search(c, head, first);

if (r1 == true) {

cout << "Value already exists" << endl;

}

else if (r == false)

{

if (p->letr > temp->letr) {

p = head;

while (p->nextPtr != head) {

p = p->nextPtr;

}

tail = p;

p->nextPtr = temp;

temp->nextPtr = head;

head->prevPtr = temp;

head = temp;

}

else if (temp->letr > p->letr) {

while (p->nextPtr != head && temp->letr > p->nextPtr->letr) {

p = p->nextPtr;

}

temp->nextPtr = p->nextPtr;

p->nextPtr = temp;

temp->prevPtr = p;

head->prevPtr = temp;

tail = temp;

}

}

}

}

/\*

Author: Nahrin Sharna

Creation Date: 03/22/2019

Modification Date: 03/25/2019

Purpose: search if the value exists in the list

\*/

bool search(char x, node \*head, node \*first)

{

/\*if (abs((head->nextPtr->letr) - x) < abs((head->prevPtr->letr) - x))

{

cout << "Forward search would be better." << endl;

}\*/

// If key is present in current node, return true

if (head->letr == x)

return true;

if (head->nextPtr == first && head->letr != x)

return false;

else {

head = head->nextPtr;

return search(x, head, first);

}

}

/\*

Author: Nahrin Sharna

Creation Date: 03/22/2019

Modification Date: 03/25/2019

Purpose: delete a value from the list

\*/

void delete1(char x) {

node \*first = head;

bool r = search(x, head, first);

string e;

try {

if (head == NULL)

throw e;

if (r == false)

throw 2;

}

catch (string e) {

cout << "List is empty." << endl;

}

catch (int f) {

cout << "Value does not exits." << endl;

}

node \*dlt; node \*curr; node \*f1 = head;

{

if (head->letr == x) {

dlt = head;

head = head->nextPtr;

delete dlt;

}

else {

curr = head;

while (curr->nextPtr->letr != x) {

curr = curr->nextPtr;

}

dlt = curr->nextPtr;

curr->nextPtr = curr->nextPtr->nextPtr;

delete dlt;

}

}

cout << "The new list is: " << endl;

do{

cout << head->letr << " ";

head = head->nextPtr;

} while (head->nextPtr != head);

}

/\*

Author: Nahrin Sharna

Creation Date: 03/22/2019

Modification Date: 03/25/2019

Purpose: Print the list in ascending order

\*/

void printListAscending(node \*head) {

node \*t = head;

try {

if (t == NULL)

throw string ("The list is empty.");

}catch (string msg) {

cout << msg << endl;

}

do{

cout << t->letr << " ";

t = t->nextPtr;

} while (t != head);

cout << endl;

}

/\*

Author: Nahrin Sharna

Creation Date: 03/22/2019

Modification Date: 03/25/2019

Purpose: Print the list in descending

\*/

void printListDescending(node \*head) {

node \*t1 = tail;

node \*t = head;

try {

if (t == NULL)

throw string("The list is empty.");

}

catch (string msg) {

cout << msg << endl;

}

do {

cout << t1->letr << " ";

t1 = t1->prevPtr;

} while (t1 != head);

cout << head->letr << endl;

# }

# Initial Test Plan:

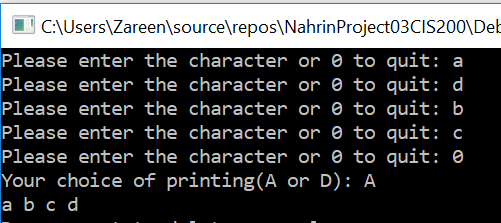
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | Insert unique characters and then print it in ascending | Char = a d b c  Choice: A | a b c d |  |  |
| 2 | Valid | Insert unique characters and then print it in ascending | Char = a d e f  Choice: D | f e d a |  |  |
| 3 | Valid | Insert same character which already exists. This will check the operation of search function as the search function will return true when the value exists. | Char = a,b,b,c | For input ‘b’ in second time:  Value already exists.  List:  a b c |  |  |
| 4 | Valid | When the list is empty, the print function(ascending) will use the exception handling to show the list is empty | Choice: A | The list is empty |  |  |
| 5 | Valid | When the list is empty, the print function(descending) will use the exception handling to show the list is empty | Choice: D | The list is empty |  |  |
| 6 | Invalid | When the option for printing is not A or D | Char = a, b, c  Choice: g | Invalid option is chosen |  |  |
| 7 | Valid | User does not want to delete any number | Delete choice:  N | Thank you for using our software |  |  |
| 8 | Invalid | When the option chosen for delete is not Y or N | Delete choice:  g | Invalid option is chosen |  |  |
| 9 | Invalid | The user wants to delete a character which does not exist in the list. This test will verify the search function as the calling search function should return false when the value does not exist. | Char = a, b, c  Delete option: Y  Character to delete: d | Value does not exist |  |  |
| 10 | Valid | User wants to delete an existing character from list | List: a b c  Delete option: Y  Value to delete: a  New list:  b c | The new list is: b c |  |  |

# Final Test Plan

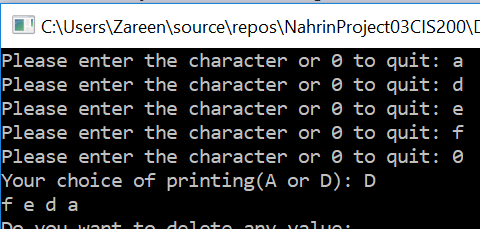
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test # | Valid / Invalid Data | Description of test | Input Value | Expected Output | Actual Output | Test Pass / Fail |
| 1 | Valid | Insert unique characters and then print it in ascending | Char = a d b c  Choice: A | a b c d | a b c d | Pass |
| 2 | Valid | Insert unique characters and then print it in ascending | Char = a d e f  Choice: D | f e d a | f e d a | Pass |
| 3 | Valid | Insert same character which already exists. This will check the operation of search function as the search function will return true when the value exists. | Char = a,b,b,c | For input ‘b’ in second time:  Value already exists.  List:  a b c | For input ‘b’ in second time:  Value already exists.  List:  a b c | Pass |
| 4 | Valid | When the list is empty, the print function(ascending) will use the exception handling to show the list is empty | Choice: A | The list is empty | The list is empty | Pass |
| 5 | Valid | When the list is empty, the print function(descending) will use the exception handling to show the list is empty | Choice: D | The list is empty | The list is empty | Pass |
| 6 | Invalid | When the option for printing is not A or D | Char = a, b, c  Choice: g | Invalid option is chosen | Invalid option is chosen | Pass |
| 7 | Valid | User does not want to delete any number | Delete choice:  N | Thank you for using our software | Thank you for using our software | Pass |
| 8 | Invalid | When the option chosen for delete is not Y or N | Delete choice:  g | Invalid option is chosen | Invalid option is chosen | Pass |
| 9 | Invalid | The user wants to delete a character which does not exist in the list. This test will verify the search function as the calling search function should return false when the value does not exist. | Char = a, b, c  Delete option: Y  Character to delete: d | Value does not exist | Value does not exist | Pass |
| 10. | Valid | User wants to delete an existing character from list | List: a b c  Delete option: Y  Value to delete: a  New list:  b c | The new list is: b c | The new list is: b c | Pass |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

# Screenshots:

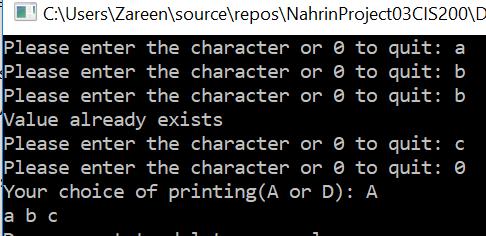
## Test Case 1:



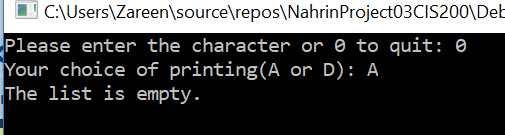
## Test Case 2:



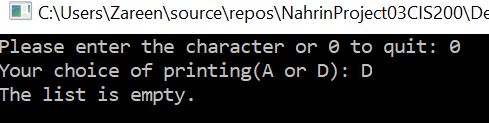
## Test Case 3:



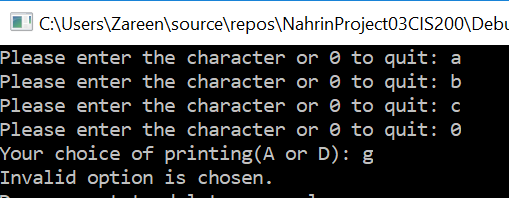
## Test Case 4:



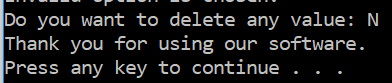
## Test Case 5:



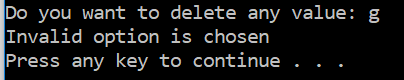
## Test Case 6:



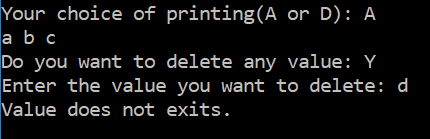
## Test Case 7:



## Test Case 8:



## Test Case 9:



## Test Case 10:

