**CSC221: DATA STRUCTURES & ALGORITHMS**

**BSCS 3*B***

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | LAB | | **08** | | Implementation of doubly linked list with the help of algorithms for Insertion, Deletion and Search an element |

|  |  |
| --- | --- |
| Objective(s): | Upon completion of this lab session, Student will be able to understand the following concepts |
| |  |  | | --- | --- | | 1 | DOUBLY LINKED LIST | | 2 | implementation of DOUBLY LINKED LIST | | 3 | Operation of DOUBLY LINKED LIST | |  |  | |  |  | | |



**Submitted By:**

* Ahsan Ghaffar
* Reg: 48411
* BS(CS)\_3B

**Submitted to:**

* Miss Ambreen Akram(AA)

**Submission Date:**

[24/04/2018]

**DEPARTMENT OF COMPUTER SCIENCE**

**BAHRIA UNIVERSITY, KARACHI CAMPUS**

|  |
| --- |
| Lab Task(s): |
|
|  |
| |  |  |  | | --- | --- | --- | | 1 | Write a program as follows   |  | | --- | | ----------------------Linked List implementation of LIST ADT-----------------------------------   1. Insert an element at the end 2. Delete an element from end 3. Search an element 4. Display List 5. Is List Empty 6. Exit   -------------------------------------------------------------------------------------------------------------  Please Enter Your Choice: | | | |

**SOURCECODE:**

#include <iostream>

using namespace std;

void insert\_end(int id);

void delete\_end();

bool search(int id);

void display();

void isempty();

int siz();

struct node {

string name;

int id;

node \*next;

node \*prev;

}; node \*head = NULL;

void insert\_end(int id)

{

node \*new1 = new node;

new1->id = id;

new1->next = NULL;

new1->prev = NULL;

node \*temp;

if (head == NULL)

{

head = new1;

}

else {

temp = head;

while (temp->next != NULL)

temp = temp->next;

temp->next = new1;

new1->prev = temp;

}

cout << "\nID is inserted at end:\n" << endl;

}

void delete\_end() {

node \*temp= head;

while (temp->next != NULL)

temp = temp->next;

temp->prev->next = NULL;

delete(temp);

cout << "\nID has been deleted from end:\n\n" << endl;

}

bool search(int id)

{

node \*temp;

temp = head;

int count = 1;

bool flag = false;

if (temp == NULL)

cout << "\nlink list underflow:\n\n" << endl;

else {

for (int i = 0; i<siz(); i++) {

if (temp->id == id) {

flag = true;

break;

}

else

temp = temp->next;

count++;

}

if (flag == true) {

cout << "\nValue found at Position: " << count << "\n\n" << endl;

return true;

}

else if (flag == false) {

cout << "\nValue not found:" << "\n\n" << endl;

return false;

}

}

}

void display() {

node \*temp = head;

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\n DISPLAY\nLink List\n\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "ID: " << temp->id << endl;

while (temp->next != NULL) {

temp = temp->next;

cout << "ID: " << temp->id << endl;

}

}

void isempty() {

node \*temp = head;

if (head == NULL)

cout << "\nLink list is Empty(Underflow)\n" << endl;

else

cout << "\nLink list is not empty:\n" << endl;

}

int siz() {

node \*temp = head;

int count = 1;

while (temp->next != NULL) {

temp = temp->next;

count++;

}

return count;

}

int main()

{

char c;

int choice, value;

do {

cout << "\n\n============================" << endl;

cout << " LINK LIST IMPLEMENTATION " << endl;

cout << "============================" << endl;

cout << "\n1- Insert an element at end " << endl;

cout << "2- Delete an element at end " << endl;

cout << "3- Search an element " << endl;

cout << "4- Display list " << endl;

cout << "5- Is list Empty " << endl;

cout << "6- Exit \n" << endl;

cout << "choose any one from the above options: ";

cin >> choice;

if (choice == 1) {

cout << "Enter an ID for insertion: ";

cin >> value;

insert\_end(value);

}

else if (choice == 2) {

delete\_end();

}

else if (choice == 3) {

cout << "Enter an ID for Searching: ";

cin >> value;

search(value);

}

else if (choice == 4) {

display();

}

else if (choice == 5) {

isempty();

}

else if (choice == 6) {

exit(0);

}

cout << "\nPress 'y' to continue and ''n' to exit" << endl;

cin >> c;

} while (c == 'Y' || c == 'y');

return 0;

}

**SCREENSHOTS:**



