Experiment 18
Student Linked List

Date: 16-11-2020

Aim: The details of students are to be stored in a linked list.

Data Structures used: Linked List

Algorithm for Searching

Input: Roll no (RN) of the student to be searched, and the Header node of the linked list
Output: A pointer to the corresponding student, if the number exists in the linked list, NULL in all other cases
Data Structure: Linked List
Steps

1. Step 1: Start
2. Step 2: ptr = Header→link //points to the first node in the list
3. Step 3: if(ptr==NULL)
4. Step 1: The linked List is empty
5. Step 2: return NULL
6. Step 4: else
7. Step 1: while(ptr!=NULL) do
8. Step 1: if(ptr→rollNo == RN) the
9. Step 1: EndWhile
10. Step 2: endif
11. Step 2: endwhile
12. Step 3: if(ptr==NULL) then
13. Step 1: return NULL
14. Step 4: else
15. Step 1: return ptr
16. Step 5: endif
17. Step 5: Stop

Algorithm for Sorting

Input: The Header Node of the Linked list to be sorted
Output : The Header node of the sorted Linked list
Data Structure : Linked List

Steps

1. Step 1: Start
2. Step 2: if(Header→link == NULL) then
3. Step 1: print("The List is empty")
4. Step 3: else
5. Step 1: temp = getNode(Node)
6. Step 2: ptr = Header→link
7. Step 3: while(Header→link!=NULL) do
8. Step 1: ptr = Header→link
9. Step 2: Header→link = ptr→link
10. Step 3: if(Header→link == NULL) then
11. Step 1: Header→link = ptr
12. Step 2: ptr→link = NULL
13. Step 4: else
14. Step 1: ptr2 = temp→link
15. Step 2: ptr1 = temp
16. while(ptr2!=NULL and ptr2→rollNo<=ptr→rollno) do

17. Step 1: ptr2 = ptr2→link
18. Step 2: ptr1 = ptr1→link
19. Step 4: endwhile
20. Step 5: ptr1→link = ptr
21. Step 6: ptr→link = ptr2
22. Step 5: endif
23. Step 4: EndWhile
24. Step 5: Header→link = temp→link
25. Step 6: returnNode(temp)
26. Step 4: endif
27. Step 5:return Header
28. Step 6: Stop

Program Code

```
/*******************************
 * Linked List Implementation
 * Done By: Rohit Karunakaran
 * *****************************/

#include<stdio.h>
#include<stdlib.h>

typedef struct Linked_List_Node
{
    struct Linked_List_Node *link;
    int rollNo;
    double mark;
    char name[40];

}Student;

void initList(Student* Header)
{
    //Header = (Student*) malloc (sizeof(Student));
    Header->link = NULL;
}

void clearList(Student **List)
{
    Student* ptr = *List;
    Student *eat = ptr;
    ptr = ptr->link;
    if(ptr!=NULL)
    {
        free(eat);
        ptr = ptr->link;
    }
}

void getStudentData(Student* node)
{
    printf("\nEnter the name of the student: ");
    scanf("%[^\n]%*c",node->name);
    printf("Enter the roll no: ");
    scanf("%d",&node->rollNo);
```

```
    printf("Enter the marks: ");
    scanf("%lf",&node->mark);
    printf("\n");
}

//Searching Algorithm
Student* searchFor(Student* Header, int rollNo)
{
    Student* ptr = Header;
    if(Header->link == NULL){
        printf("The List is Empty\n");
        return NULL;
    }
    else
    {
        while(ptr!=NULL)
        {
            if(ptr->rollNo == rollNo)
            {
                return ptr;
            }
            ptr = ptr->link;
        }
        return NULL;
    }
}

//Sorting algorithm
void sortStudentList(Student** Header)
{
    if((*Header)->link==NULL)
    {
        printf("The List is empty]\n");
    }
    else
    {
        Student *temp =(Student*) malloc(sizeof(Student));
        Student *ptr=NULL;
        temp->link=(*Header)->link;
        (*Header)->link = NULL;

        while(temp->link!=NULL)
        {
            ptr = temp->link;
            temp->link = ptr->link;
            if((*Header)->link ==NULL)
            {
                (*Header)->link = ptr;
                ptr->link = NULL;
            }
            else
            {
                Student *ptr2=(*Header)->link;
                Student *ptr1 =(*Header);
                while(ptr2!=NULL && ptr2->rollNo<=ptr->rollNo)
                {
```

```
                    ptr2=ptr2->link;
                    ptr1=ptr1->link;
                }
                ptr1->link=ptr;
                ptr->link = ptr2;

            }

        }
        free(temp);
    }
}

void dispStudent(Student* ptr)
{
    printf("\nName: %s",ptr->name);
    printf("\nRoll No: %d",ptr->rollNo);
    printf("\nMarks: %lf",ptr->mark);
}

//Insertion Algorithms
void insertStart(Student *Header)
{
    Student *new_node = (Student*) malloc(sizeof(Student));

    if(new_node!=NULL)
    {
        getStudentData(new_node);
        new_node->link = NULL;
        Student* ptr = Header->link;
        Header->link = new_node;
        new_node->link=ptr;
    }
    else
    {
        printf("Insertion Not Possible\n");
        exit(1);
    }
    return ;
}


void deletionAt(Student* Header, int rollNo)
{
    if(Header->link == NULL)
    {
        printf("Deletion not possible. The list is empty\n");
    }
    else
    {
        Student* ptr = Header;
        while(ptr->link!=NULL)
        {
            if(ptr->link->rollNo==rollNo)
                break;
            ptr=ptr->link;
```

```
        }
        if(ptr->link!=NULL)
        {
            Student* red = ptr->link;
            ptr->link = ptr->link->link;
            printf("The Student to be deleted is :\n");
            dispStudent(red);
            free(red);
        }
        else
        {
            printf("The Given RollNo is not found \n");
        }
    }
}



void displayList(Student* Header)
{
    Student* ptr = Header->link;
    if(ptr!=NULL)
    {

        while(ptr!=NULL)
        {
                  printf("\n");
            dispStudent(ptr);
            printf("\n");
            ptr=ptr->link;
        }
        printf("\n");
    }
    else
    {
        printf("The Linked list is empty\n");
    }
}

int menu(Student* Header)
{
    int RUN = 1;
    while(RUN)
    {
        printf("\n");
        printf("============================\n");
        printf("            MENU            \n");
        printf("============================\n");
        printf("1.Insert\n");
        printf("2.Delete Student\n");
        printf("3.Display the linked List\n");
        printf("4.Search for a Student by Roll No\n");
        printf("5.Sort By Roll No\n");
        printf("6.Exit\n");
        printf("Enter Choice: ");
        int choice;
```

```
int pos;
scanf("%d%*c",&choice);

switch(choice)
{
    case 1:
            insertStart(Header);
            printf("\n");
            break;

    case 2: printf("Enter the roll no of the student to be deleted : ");
            scanf("%d%*c",&pos);
            deletionAt(Header,pos);
            printf("\n");
            break;

    case 3: printf("\nThe Student List is : ");
                    displayList(Header);
            break;

    case 4: printf("Enter the roll Number to be searched for : ");
            scanf("%d%*c",&pos);
            Student* res = searchFor(Header,pos);
            if(res == NULL)
            {
                printf("The given roll number is invalid !!!\n");
            }
            else
            {
                dispStudent(res);
            }
            break;

    case 5: sortStudentList(&Header);
                    printf("The sorted list is :\n");
                    displayList(Header);
            break;

    case 6: RUN=0;
            break;

    default: printf("Enter a valid choice\n");
```

```
                    printf("\n");
                    break;

            }
        }
        printf("Exiting........\n");
        clearList(&Header);
        return RUN;
    }


    int main(){
        Student *Header = (Student*)malloc(sizeof(Student));
        initList(Header);
        return menu(Header);
    }
```

Sample input output

```
..ograming/C/CSL201/2020-11-16❱ ./StudentLinkedList.o

============================
          MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 1

Enter the name of the student: Helen
Enter the roll no: 28
Enter the marks: 87



============================
          MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 1

Enter the name of the student: Abhiram
Enter the roll no: 7
Enter the marks: 89



============================
          MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 1

Enter the name of the student: Rajmohan
Enter the roll no: 43
Enter the marks: 89
```

```
============================
          MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 5
The sorted list is :


Name: Abhiram
Roll No: 7
Marks: 89.000000


Name: Helen
Roll No: 28
Marks: 87.000000


Name: Rajmohan
Roll No: 43
Marks: 89.000000


============================
          MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 4
Enter the roll Number to be searched for : 28

Name: Helen
Roll No: 28
Marks: 87.000000
```

```
============================
            MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 2
Enter the roll no of the student to be deleted : 7
The Student to be deleted is :

Name: Abhiram
Roll No: 7
Marks: 89.000000

============================
            MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 3

The Student List is :

Name: Helen
Roll No: 28
Marks: 87.000000


Name: Rajmohan
Roll No: 43
Marks: 89.000000


============================
            MENU
============================
1.Insert
2.Delete Student
3.Display the linked List
4.Search for a Student by Roll No
5.Sort By Roll No
6.Exit
Enter Choice: 6
Exiting........
```