DATE:13.01.15

**ASSIGNMENT NO.19**

->PROBLEM STATEMENT:

Write a program in C for the following operation pertaining to a single linked list where each node contains one number.

1. Create the list.
2. Find the sum and average and standard deviation of elements in the list.
3. Sort the list.
4. Find the maximum and minimum of the list.

**->ALGORITHM:**

**Data Structure Definition**

[ The data structure Node represents a single node in the linked list]

*Structure Node*

*( data* to hold the relevant data value in a Node,

*next* that holds the address of the next Node

*)*

1. **Algorithm for function *createlist()*.**

Purpose : To create a single linked list.

Step 1:- [ Initialize the ‘head’ pointer]

head:= NULL.

Step 2:- Input item [such that item ≠ -999]

Step 3:- While ( item ≠ -999)

1. Allocate memory for a new Node and Assign its pointer as new\_node
2. new\_node → data := num
3. new\_node → next := NULL
4. If ( head = NULL) Then
5. head := new\_node
6. Else
7. ptr → next := new\_node

[ end of If…..then…..else……structure]

1. ptr := new\_node
2. Input item [such that item ≠ -999]

[ end of while….. loop]

Step 4:- Return head

**(2) Algorithm for function *lismatht().***

Purpose :- To find sum, average and standard deviation of elements of the list.

Step 1:- [Initialize the variables]

1. sum := 0
2. average := 0
3. count := 0
4. sum2 := 0

Step 2:- ptr := head

Step 3:- While (ptr ≠ NULL)

1. sum := sum + ptr→data
2. ptr := ptr→next
3. count := count + 1

[End of While……loop]

Step 4:- average := (sum / count)

Step 5:- ptr := head

Step 6:- While (ptr ≠ NULL)

1. sum2 := sum2 + pow((ptr→data) – average), 2);
2. ptr := ptr→data

[End of While…..loop]

Step 7:- variance := (sum2 / count)

Step 8:- std\_deviation := sqrt(variance)

Step 9:- Display “sum”, “average” and “std\_deviation”.

Step 10:- Return head.

**(3) Algorithm for function *sortlist().***

Purpose:- To sort a linked list in ascending using bubble sort technique.

Step 1:- [Check for existence of the list]

If (head = NULL) then

1. Display “List not exist”.
2. Exit.

[End of If…….then……structure]

Step 2:- ptr2 := head

Step 3:- While (ptr2 ≠ NULL)

1. ptr1 := ptr2→next.
2. While (ptr1 ≠ NULL)
3. If (ptr2→data > ptr1→data) then
4. temp := ptr2→data
5. ptr2→data := ptr1→data
6. ptr1→data := temp

[End of If…..then……structure]

1. ptr1 := ptr1→next.

[End of While…….loop]

1. ptr2 := ptr→next.

[End of While……..loop]

Step 4:- Return head.

**(3) Algorithm to find the maximum and minimum element of the list.**

Purpose:- To find the maximum and minimum element from the list.

\Step 1:- [Check for existence of the list]

If (head = NULL) then

1. Display “List not exist.”
2. Exit.

[End of If….then…..structure]

Step 2:- ptr ← head

Step 2:- Display “ptr→data” as minimum element of the list.

Step 3:- While (ptr→next ≠ NULL)

1. ptr := ptr→next

[End of While…..loop]

Step 4:- Display “ptr→data” as maximum element of the list.

Step 5:- Return head.

->CODE:

/\* PROGRAM FOR OPERATIONS OF SINGLY LINKED LIST \*/

# include<stdio.h>

# include<conio.h>

# include<stdlib.h>

# include<math.h>

typedef struct node

{

int data;

struct node \*next;

}node;

struct node \*head;

struct node \*getnode(int x)

{

node \*p;

p = (node\*)malloc(sizeof(node));

p->data = x;

p->next = NULL;

return p;

}

struct node \*createlist()

{

node \*new\_node,\*ptr;

int item;

printf("\nEnter data to insert (TYPE -999 to STOP) : ");

scanf("%d",&item);

while (item != -999)

{

new\_node = getnode(item);

if (head == NULL)

head = new\_node;

else

ptr->next = new\_node;

ptr = new\_node;

printf("Enter data to insert (TYPE -999 to STOP) : ");

scanf("%d",&item);

}

return head;

}

void displaylist()

{

node \*ptr;

if (head == NULL)

{

printf("\nList not exist. Create list first.");

getch();

exit(0);

}

printf("\nElements of the list are : \n");

ptr = head;

while (ptr != NULL)

{

printf("%d-->",ptr->data);

ptr = ptr->next;

}

printf("List ends here.");

}

struct node \*listmath()

{

node \*ptr;

int sum=0,sum2=0,count=0;

float ave,std\_deviation,variance;

if (head == NULL)

{

printf("\nList not exist. Create list first.");

getch();

return head;

}

ptr = head;

while (ptr != NULL)

{

sum = sum + ptr->data;

count++;

ptr = ptr->next;

}

printf("\n\nSum of elements of list : %d",sum);

ave =sum / count;

printf("\n\nAverage of elements of list : %f",ave);

ptr = head;

while (ptr != NULL)

{

sum2 = sum2 + pow((ptr->data-ave),2);

ptr = ptr->next;

}

variance = (sum2 / count);

std\_deviation = sqrt(variance);

printf("\n\nStandard deviation of elements of list : %f",std\_deviation);

getch();

return head;

}

struct node \*sortlist()

{

node \*ptr1,\*ptr2;

int temp;

if (head == NULL)

{

printf("\n\nList not exist. Create list first.");

getch();

return head;

}

ptr2 = head;

while (ptr2 != NULL)

{

ptr1 = ptr2->next;

while (ptr1 != NULL)

{

if (ptr2->data > ptr1->data)

{

temp = ptr2->data;

ptr2->data = ptr1->data;

ptr1->data = temp;

}

ptr1 = ptr1->next;

}

ptr2 = ptr2->next;

}

return head;

}

struct node \*maxlist()

{

node \*ptr;

if (head == NULL)

{

printf("\n\nList not exist. Create list first.");

getch();

return head;

}

ptr = head;

printf("\n\nMinimum element of the list : %d",head->data);

while (ptr->next != NULL)

ptr = ptr->next;

printf("\n\nMaximum element of the list : %d",ptr->data);

getch();

return head;

}

void main()

{

int ch;

clrscr();

printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t WELCOME");

printf("\n This is a menu-drivven program for operations of singly linked list.");

getch();

do

{

clrscr();

printf("1. Create list.");

printf("\n2. Display list.");

printf("\n3. Find sum, average and dtandard deviation.");

printf("\n4. Sort the list.");

printf("\n5. Find the maximum and minimum of the list.");

printf("\n6. EXIT from program.");

printf("\n\nEnter your choice : ");

scanf("%d",&ch);

switch(ch)

{

case 1:

createlist();

break;

case 2:

displaylist();

getch();

break;

case 3:

listmath();

getch();

break;

case 4:

sortlist();

getch();

break;

case 5:

maxlist();

break;

case 6:

clrscr();

printf("\n\n\n\n\n\n\n\n\n\n\n\t\t\t\t THANK YOU");

getch();

exit(0);

default:

printf("\n\nYou enter a wrong choice. Enter correct choice.");

getch();

break;

}

}while (1);

}

**->OUTPUT :**

/\* ================ OUTPUT ==================

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 2

List not exist. Create list first.1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 6

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 1

Enter data to insert (TYPE -999 to STOP) : 3

Enter data to insert (TYPE -999 to STOP) : 4

Enter data to insert (TYPE -999 to STOP) : 1

Enter data to insert (TYPE -999 to STOP) : 5

Enter data to insert (TYPE -999 to STOP) : 6

Enter data to insert (TYPE -999 to STOP) : -999

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 2

Elements of the list are :

3-->4-->1-->5-->6-->NULL.

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 3

Sum of elements of list : 19

Average of elements of list : 3.000000

Standard deviation of elements of list : 1.732051

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 4

List before sorting :

Elements of the list are :

3-->4-->1-->5-->6-->NULL.

List after sorting :

Elements of the list are :

1-->3-->4-->5-->6-->NULL.

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 5

Minimum element of the list : 1

Maximum element of the list : 6

1. Create list.

2. Display list.

3. Find sum, average and standard deviation.

4. Sort the list.

5. Find the maximum and minimum of the list.

6. EXIT from program.

Enter your choice : 6

\*/

**->DISCUSSION:**

Linked list is a example of dynamic memory allocation. List refers to a set of items organized sequentially. A completely different way to represent a list is to make each item in the list part of a structure that also contains a ‘next’ part that keeps the address of successor node. This type of lit is called linked list because it is a list whose order is given by links from one item to the next.

Here we consider a linked list whose every node has two parts-

1. “Data” part , which keeps some data.
2. “Next” part, which keeps the address of the next node.

Here we used a pointer ‘head’ which keeps the address of the first node of the linked list. Initially we consider the value of the pointer is NULL. Then we do all the insertion and deletion operations of linked list.