Lab Report 03

Singly linked list

Submitted by

Muhammad Awais / 2883

Submitted To

Ma'am Zanaib Malik

Department of Computer Science and Engineering

National University of modern language, Islamabad

List of Task

1. Concepts of linked list
2. Define the requirements to used linked list.
3. Discuss the memory management of linked list.
4. Describe Limitation of array which leads the usage of linked list.
5. Made Class Node // Header file
6. Made Singly Linked list Class// Header file
7. Made main class // cpp
8. Insert Node
9. Add to Head
10. Add To Tail
11. Traverse Function
12. Use of Generic variable which make a node to store all variable.

Description of Node Class

```
/*Class Node which store the information and address of next node in variable t
Data and t Next*/
#include <iostream>
using namespace std;
template<class t> // Generic Data Type
class Node
{
private:
    t data;// info
    Node<t> *next;// address of next node
```

What does Class Node Do?

1. We made a header file named Class Node (Node.h) which store the Data on nodes and address of next node.
2. Implement the **template<class t>** which makes the data types of and Data and next as a generic.
3. In Node class we private the Data and Next to secure the information of each Node in linkedlist.

4. We used Setter to set the values on private instantaneous variable (Data, *next).
5. Getter returns the set values in Data and Next.
6. To display the Data and next address on node we used Void Display () which display the address of node, data on node and address of next node.

Source Code of Class Node (Node.h)

```
/*Class Node which store the information and address of next node in variable t
Data and t Next*/
#include <iostream>
using namespace std;
template<class t> // Generic Data Type
class Node
{
private:
    t data;// info
    Node<t> *next;// address of next node

public:
    Node(int data, Node *next)
    {
        this->data = data;
        this->next = next;
    }

    void setData(t data);
    void setNext(Node<t> *next);

    t getData();
    Node<t> *getNext();

    void display();

}; // End of class Node

template<class t>
void Node<t>::setData(t data) //::scope resulution operator

{
    this->data = data;
} // End of setData
```

```
template<class t>
void Node<t>::setNext(Node<t> *next)
{
    this->next = next;
} // End of setnext

template<class t>
t Node<t>::getData()
{
    return this->data;
} // End

template<class t>
Node<t>* Node<t>::getNext()
{
    return this->next;
} // End

template<class t>
void Node<t>::display()
{
    cout << "(" << this << ") |" << data << "|" << next << "|" << endl;
} // End
```

Output

Node.h

```
/*Class Node which store the information and address of next node in variable t Data and t Next*/
#include <iostream>
using namespace std;
template<class t> // Generic Data Type
class Node
{
private:
    t data;// info
    Node<t> *next;// address of next node

public:
    Node(int data, Node *next)
    {
        this->data = data;
        this->next = next;
    }

    void setData(t data);
    void setNext(Node<t> *next);

    t getData();
    Node<t> *getNext();
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\user\OneDrive\Desktop\Node.h.gch
- Output Size: 6.67514419555664 MiB
- Compilation Time: 0.34s
```

Description of Singly linked list Class

```
/*In SLL class g=first we include our user made Header file -> Node.h. In SLL
class we Declared and
defined required function with passage of time. SLL class perform all methods and
implementation
of linked list*/
#include <iostream> // Standard Headerfile
#include "Node.h" // user define headerfile
using namespace std;
template<class t>  // It's enable the generic variable
class SLL
{
private:
    Node<t> *head; // first Node
    Node<t> *tail; // last Node
```

What does class SLL Do?

1. SLL class initialize the value of Head and Tail using constructor.
2. Made Data and next as private for security.
3. Use Getter and Setter to give and get values of Data and Next.
4. Using SLL class, we add node on different point by using Function.

Source Code

```
/*In SLL class g=first we include our user made Header file -> Node.h. In SLL
class we Decalared and
defined required function with passage of time. SLL class perform all methods and
implementation
of linkedlist*/
#include <iostream> // Standard Headerfile
#include "Node.h" // user define headerfile
using namespace std;
template<class t>  // It's enable the generic variable
class SLL
{
```

```
private:
    Node<t> *head; // first Node
    Node<t> *tail; // last Node

public:
    SLL()
    {
        head = 0;
        tail = 0;
    }

    //************** Function Decalarations ****************

    void setHead(Node *first);
    void setTail(Node *last);
    Node *getHead();
    Node *getTail();
//**************************** Functions Definations **********************

template<class t>
void SLL<t>::setHead(Node<t> *first)
{
    head = first;
} // End of Set Head

template<class t>
void SLL<t>::setTail(Node<t> *last)
{
    tail = last;
} // End of Set Tail

template<class t>
Node<t>* SLL<t>::getHead()
{
    return head;
} // End of Get Head

template<class t>
Node<t>* SLL<t>::getTail()
{
    return tail;
} // End of Get Tail
```

Description of Void Traverse

```
void traverse(); // vist all
```

What does Function Do?

1. Using loop
2. Make a pointer ptr start from head and terminate the loop when head=NULL mean on tail and ptr move by storing the next node address.
3. Display the Node, Data and Address of next node.

Source Code

```
#include <iostream>
#include "SLL.h"
using namespace std;

int main(int argc, char **argv)
{

  SLL<int> list1;
  list1.addToHead(17); // /17/0 H,T
  list1.addToHead(15); // without new mean compile time call // /15\0xA1/H
  list1.addToHead(19);
  list1.traverse();
  //  //19,15,17
```

```
template<class t>
void SLL<t>::traverse()
{
    Node<t> *n = head;

    while (n != NULL)
    {
        n->display();
        //cout << n->getData()<<endl; // display Data of N
        n = n->getNext();     // get address of next
    }

} // End of traverse
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS E:\BSCS 3rd Semester\DSA\My_code\Singular linklist> cd "e:\BSCS 3rd Semester\DSA\My_code\Singular l
 main.cpp -o main } ; if ($?) { .\main }
(0x1081490) |19|0x1081480|
(0x1081480) |15|0x1081470|
(0x1081470) |17|0|
PS E:\BSCS 3rd Semester\DSA\My_code\Singular linklist>
```

Description of Void Add to Head

```
void addToHead(t element);
```

What does Function Do?

1. Create an object n class of Node which store information and address of nodes.
2. It checks five Possible mapping.
3. Head and tail zero (list empty) then add node on head which also become tail.
4. If there at least one or more node in the list, then set head on the next of new node and made node as a head.

Source Code

```
#include <iostream>
#include "SLL.h"
using namespace std;

int main(int argc, char **argv)
{

  SLL<int> list1;
  list1.addToHead(17); // /17/0 H,T
```

```
  list1.addToHead(15); // without new mean compile time call // /15\0xA1/H
  list1.addToHead(19);
  list1.traverse();
  //  //19,15,17
```

```
template<class t>
void SLL<t>::addToHead(t element)
{
    /* Check 5 Maping Possibilities

     1- Error -> No
     2- Only Head modify -> Yes b/c we add on head so position change
     3- Only Tail modify -> No
     4- Head and Tail both modify -> No
     5- Nor Head not Tail Modify-> No
     */

    Node<t> *p = new Node<t>(element, NULL);

    if (head == 0 && tail == 0) // empty
    {
        head = p;
        tail = p;
    }

    else // Not empty
    {
        p->setNext(head);
        head = p;
    }

} // End of Add To Head
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS E:\BSCS 3rd Semester\DSA\My_code\Singular linklist> cd "e:\BSCS 3rd Semester\DSA\My_code\Singular l
 main.cpp -o main } ; if ($?) { .\main }
(0x1081490) |19|0x1081480|
(0x1081480) |15|0x1081470|
(0x1081470) |17|0|
PS E:\BSCS 3rd Semester\DSA\My_code\Singular linklist>
```

Description of Void Add to Tail

```
void addToTail(t element);
```

What does Function Do?

1. Create an object n class of Node which store information and address of nodes.
2. It checks five Possible mapping.
3. Head and tail zero (list empty) then add node on tail which also become head.
4. If there at least one or more node in the list, then set new node on the next of tail and made new node as a tail.

Source Code

```
#include <iostream>
#include "SLL.h"
using namespace std;

int main(int argc, char **argv)
{

  SLL<int> list1;
  // list1.addToHead(17); // /17/0 H,T
  // list1.addToHead(15); // without new mean compile time call // /15\0xA1/H
  // list1.addToHead(19);
  //list1.traverse();
  //  //19,15,17
  list1.addToTail(101);
```

```
  list1.addToTail(99);
  list1.addToTail(29);
  list1.traverse();

  SLL<char> list2;

  list2.addToTail('X');
  list2.addToTail('A');
  list2.addToTail('Z');
  list2.traverse();
  return 0;
}
template<class t>
void SLL<t>::addToTail(t element)
{   /* Check 5 Maping Possiblities
    1- Error -> No
    2- Only Head modify -> No
    3- Only Tail modify -> Yes b/c we add on tail so position change
    4- Head and Tail both modify -> No
    5- Nor Head not Tail Modify-> No
    */
    Node<t> *ptr = new Node<t>(element, NULL);
    if (head == 0 && tail == 0) // empty
    {
        head = ptr;
        tail = ptr;
    }
    else // Not empty
    {
        tail->setNext(ptr);
        tail = ptr;
    }
} // End of Add To Tail
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

PS E:\BSCS 3rd Semester\DSA\My_code\Singular linklist> cd "e:\BSCS 3rd Semester\DSA\My_code\Singular li
 main.cpp -o main } ; if ($?) { .\main }
(0xfb1470) |101|0xfb1480|
(0xfb1480) |99|0xfb1490|
(0xfb1490) |29|0|
(0xfb13d8) |X|0xfb13e8|
(0xfb13e8) |A|0xfb13f8|
(0xfb13f8) |Z|0|
PS E:\BSCS 3rd Semester\DSA\My_code\Singular linklist>
```

End of Lab 03