



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

## Developing all Possible Library Function on Linked List to Enhancement of C Compiler

Sudip Sinha<sup>1</sup>, Pritam Ghosh<sup>1</sup>, Prof. (Dr.) Pranam Paul<sup>2</sup>

MCA Final Year Student, Narula Institute of Technology, Agarpara, Kolkata, West Bengal, India<sup>1</sup>

HOD, Department of Computer Application, Narula Institute of Technology, Agarpara, Kolkata, West Bengal, India<sup>2</sup>

**ABSTRACT:** In this paper, we try to add some user define functions in C compiler like printf() , scanf(). Different operations of linked list, which are not present in C – Compiler, are being tried add as library functions. Then to implemented of different kinds of linked list operations can directly be used as calling of functions which is written in byte code. Header file of these functions has also been generated though writing the prototype. Our goal is adding more features in c compiler to make more useful for any programmer, but with maintaining proper concept and process of compiling the code through C compiler.

**KEYWORDS:** First Node Insert, Last Node Insert, Add specific last Node Insert, Add specific previous Node Insert, Delete Node, Sorting value, Display value, C- Compiler, Library File, Header File, Byte Code, Object File. Linker, Loader,

### I. INTRODUCTION

Queues and Stacks share two traits; they both have strict rules for accessing the data stored in them, and the retrieval operations are, by nature, consumptive. In other words, accessing an item in a stack or queue requires its removal, and unless the items stored elsewhere, it's destroyed. Unlike a stack or a queue, a *linked list* can be accessed in a flexible fashion, because each piece of information carries with it a link to the next data item in the chain. In Addition, a linked list retrieval operation does not remove and destroy an item from the list. In fact we need to add a specific deletion operation to do this.

### II. RELATED WORK

In many times, we code in C – compiler to implement different type of coding for keeping huge number of data in RAM, but number of data is not known to us before the keep the data. Number of data is increased during the run time. According to the requirement to keep the data it will be kept by maintaining link with of set of these elements. So we have to code linked list to implement the program. If this is already exist in C – compiler, it is beneficial to the coder for not to write coding for linked list, but only calling the functions.

We are implemented the *linked list*. Here we created library file with those mathematical expression, and corresponding prototype is declared in the header file. In our project we use this lib file and header file for matrix operation. Here we give our own header file and call the function from main (). We used to create .obj file and .lib file for linked list using tcc and tlib command. In this paper, adding note at first, at last, at specific place, deleting the node, sorting the data of linked list, etc. is properly implemented with proper concept of C - Compiler.

### III. PROPOSED METHODOLOGY

**3.1 Defining the Structure of a Node:** Here we use structure which have two type of variable one of them is used for value of the Node, another one is address of the Node . Here we worked with one value of one corresponding node.

```
struct ll
{
    data_type v;
    struct ll*a;
```



## International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
};
```

**3.2 Create node at Begin:** Here we create a node at 1<sup>st</sup> where the address value is hold by head. In the following the function we have written. Here 2<sup>nd</sup> argument is use for take the value of the node that's address is hold by head, which is indicated by 1<sup>st</sup> argument

```
int addbgn_node(struct ll**h, int n)
{
    struct ll *p,*q;
    p=(struct ll*)malloc(sizeof(struct ll));
    if(p==NULL) return(0); // return 0 if memory is full. No new node will be added.
    p->v =n;
    p->a = NULL;
    p->a = *h;
    *h=p;
    return(1); // return 1 for successfully append
}
```

**3.3 Add Node at last:** Here we add a node at the end of the linked list. In the following the function we have written. Here 2<sup>nd</sup> argument is use for take the value of the node which is added end of the previous node whose address is hold by head of linked list which is indicated by 1<sup>st</sup> argument.

```
int add_lastnode(struct ll**h ,data_type n)
{
    struct ll *p,*q;
    p=(struct ll*)malloc(sizeof(struct ll));
    if(p==NULL) return(0); // return 0 if memory is full. No new node will be added.
    p->v =n;
    p->a = NULL;
    if(*h != NULL)
    {
        q= *h;
        while(q->a!=NULL)
        {
            if(q->v==n) break;
            q= q->a;
        }
        q->a=p;
    }
    else
        *h=p;
    return(1); // return 1 for successfully append
}
```

**3.4 Add node at specific place:** Here we add note at any position of the linked list, if we want to add new node at the previous of the specific node then we can do this, if we want to add new node at the last of the specific node then we can do this. In the following the function we have written. Here 1<sup>st</sup> argument is use for the head which is hold the address value of the 1<sup>st</sup> Node. 2<sup>nd</sup> argument is use for input value of the node which is added after or before a specific node, this specific node is indicated by 3<sup>rd</sup> argument. But in the case of 2<sup>nd</sup> function, addspecificprevious\_node() we use one extra argument which is used for store previous address value .

```
int addspecificlast_node(struct ll**h,intn,int s)
{
    struct ll *p,*q,*t;
    p=(struct ll*)malloc(sizeof(struct ll));
    if(p==NULL) return(0); // return 0 if memory is full. No new node will be added.
    p->v =n;
```



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
p->a = NULL;
if(*h != NULL)
{
    q= *h;
    while(q->a!=NULL)
    {
        if(q->v==n) break;
        q= q->a;
        if(q->v==s) t=q;
    }
    p->a=t->a;
    t->a=p;
}
else
    *h=p;
return(1); // return 1 for successfully append
}

int addspecificprevious_node(struct ll**h, int n, int s, int i)
{
    struct ll *p,*q,*t;
    p=(struct ll*)malloc(sizeof(struct ll));
    if(p==NULL)
    {
        free(p);
        return(0); // return 0 if memory is full. No new node will be added.
    }
    p->v =n;
    p->a = NULL;
    if(*h != NULL)
    {
        q= *h;
        while(q->a!=NULL)
        {
            if(q->v==n) break;
            q=q->a;
            if(q->v==s) t=q;
        }
        p->a=t->a;
        t->a=p;
        i=p->v;
        p->v=t->v;
        t->v=i;
    }
    else
        *h=p;
    return(1); // return 1 for successfully append
}
```

**3.5 Delete Node:** Here we delete any node from any here . In the following the function we have written. Here 1<sup>st</sup> node is use for head which is hold address of the 1<sup>st</sup> node of linked list, 2<sup>nd</sup> argument is use for particular delete the node.

```
int delete_node(struct ll**h, int s)
```



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
{
    struct ll *q,*t;
    q=*h;
    while(q->a!=NULL)
    {
        if(q->v==s) break;
        t=q;
        q=q->a;
    }
    if(q==*h) *h=(*h)->a;
    else
    {
        t->a=q->a;
        free(q);
        return(1);    // return 1 deleted value successfully
    }
}
```

**3.6 Sorting the Value:** Here we sorting the value after inserting any position. In the following the function we have written. Here 1<sup>st</sup> argument is use hold the address of the 1st node of linked list

```
int sorting_value(struct ll*h)
{
    data_types wapped , i, temp;
    struct ll *q;
    struct ll *t=NULL;
    do
    {
        swapped=0;
        q=h;
        while(q->a!=t)
        {
            if(q->v > q->a->v)
            {
                temp=q->a->v;
                q->a->v=q->v;
                q->v=temp;
                swapped=1;
            }
            q=q->a;
            t=q;
        }
        while(swapped);
    }
}
```

**3.7 Display the value:** Here we displayed all entered value .In the following the function we have written of the 1st node of linked list

```
int display_value(struct ll*h)
{
    q=h;
    while(q!= NULL)
    {
        printf("\t%d",q->v);
    }
}
```

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
        if(q->a == NULL ) break;
        q=q->a;
    }
    return(1); // return 1 display the value successfully
}
```

### 3.8 Using command to create .obj file and .lib file :

Create the .obj file for the linked list by using tcc.exe

Step 1. At first we write the code of linked list in notepad and save as it .C extension.

Step 2. Then go to the dos prompt and write the command `tcc -c filename.c`

Ex. `tcc -c linklist.c`

Create the .lib file for the particular function of the linked list by using tlib.exe

Step1. At first we write the only function for the linked list operation because lib file contains only function body.

Step2. Then go to the dos prompt and write the command `tlib /C mylib.lib+filename.obj`

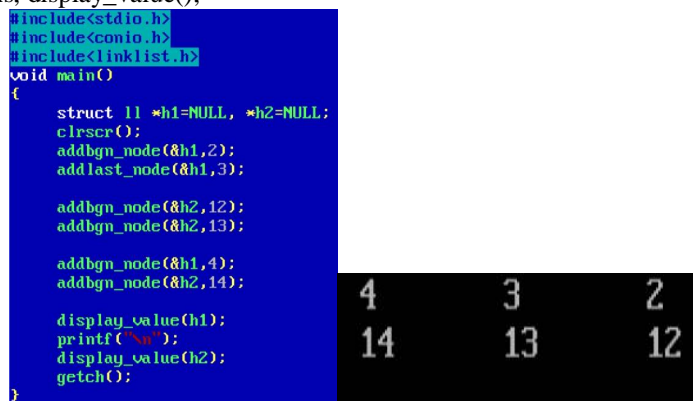
Ex. `.tlib /C mylib.lib + linklist.obj`

Header file –We write down the only prototype of the functions of linked into a .h extension file, named linklist.h.

Example: `#include<linklist.h>`

## IV. EXAMPLE OF EXECUTION AND RESULT

Here we are giving some example of our success fully created library functions. In Black screen (Right Side) of the all figures are shown the output of the executed program which are also shown others screen (Left side) of the corresponding figures. Figure 4.1 shows the example of Create node at 1<sup>st</sup> in linked list whereas in Figure 4.2, way of calling functions for Create Add Node at last in linked list is shown with its successful output of execution of code 3.3. The Sorting of linked list is being focused in figure 4.4. All though, displaying the linked list is respectively done through calling of functions, `display_value()`;



```
#include<stdio.h>
#include<conio.h>
#include<linklist.h>
void main()
{
    struct ll *h1=NULL, *h2=NULL;
    clrscr();
    addbgn_node(&h1,2);
    addlast_node(&h1,3);

    addbgn_node(&h2,12);
    addbgn_node(&h2,13);

    addbgn_node(&h1,4);
    addbgn_node(&h2,14);

    display_value(h1);
    printf("\n");
    display_value(h2);
    getch();
}
```

4	3	2
14	13	12

Figure 4.1

Coding for calling of Create node at 1<sup>st</sup> in linked list and output

In the figure 4.1, two linked list had been created with head h1 and h2. In the this program, it is clearly shown in this program that 2, 3, 4 have been instated into h1 and 12, 13, 14 have been inserted in the h2 linked list by only calling the function `addbng_node()`.

# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```

#include<stdio.h>
#include<conio.h>
#include<linklist.h>
void main()
{
    struct l1 *h1=NULL, *h2=NULL;
    clrscr();
    addlast_node(&h1,2);
    addlast_node(&h1,3);

    addlast_node(&h2,12);
    addlast_node(&h2,13);

    addlast_node(&h1,4);
    addlast_node(&h2,14);

    display_value(h1);
    printf("\n");
    display_value(h2);
    getch();
}

```

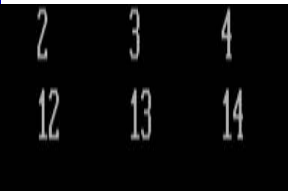


Figure 4.2

Coding for calling of Create Add Node at last in linked list and output

In the figure 4.2, two linked list had been also created with head h1 and h2. In the this program, it is clearly shown in this program that 2, 3, 4 have been instated into h1 and 12, 13, 14 have been inserted in the h2 linked list by only calling the function addlast\_node().

```

#include<stdio.h>
#include<conio.h>
#include<linklist.h>
void main()
{
    struct l1 *h1=NULL;
    clrscr();
    addlast_node(&h1,20);
    addbgn_node(&h1,23);
    addbgn_node(&h1,14);
    printf("\nBefore Shorting");
    display_value(h1);

    sorting(h1);
    printf("\nAfter Shorting");
    display_value(h1);

    getch();
}

```



Figure 4.3

Coding for calling of Create Sorting the Value in linked list and output

In the figure 4.3, in the this program, it is clearly shown in this program that 14, 23, 20 have been instated into h1 and after shorting, it will 14, 20, 23 by only calling the function addlast\_node().

## V. ANALYSIS

Here we use structure which has two type of variable one of them is used for value of the Node; another one is address of the Node. Here we have worked with one data value of one corresponding node. Here head holds the address the value of the 1<sup>st</sup> node.

In these proposed functions, pointer to a pointer is being passed as argument which used to receive the address of head, When within the function to change the value of a variable which is not defined into that of the function it is only possible if we pass address of the variable in c. Here our proposed library function is kept into the library file. But the head of the linked list is defined with the programs which are totally outside of the library file. Head to change the value of the head according to our requirement might be null or the address of the first node we have to pass address of the head within our proposed library function. As head in itself a pointer so address of the head should be caught by the pointer to pointer. So in the argument list this pointer is needed to define.



# International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

In the time of displaying the linked list through function `display_value()`, address of head is not being passed, because in this function, value of head is not need to change, but only display the values of the entire linked list.

Head (h) stores the address of pointer q (&q) , \*h stores the address stored by pointer q and \*\*h stores the value at address stored by q. In simple language `h=&q`, `*h= q` and `**h= *q`. if we write `*h= 1528` then it means that value at address stored in h becomes 1528 and since address stored in h is the address of pointer q (&q) thus now `q=1528` (i.e. address stored in q is 1528) and this change is permanent. Whenever we are changing value of \*h we are indeed changing value at address stored in h and since `h=&q` (address of pointer q) we are indirectly changing value of q or address stored in q.

## VI. CONCLUSION

In this enhanced compiler, it is clearly noddod that without writing the coded for following the linkedlist operation, programmer can code by only calling of the functions of those operations. Function – prototypes of all these functions have been written in a header file, named `linklist.h` which is only needed to include for calling the these functions.

Insert first node of the linked list	:	by the function, <code>add_1stnode()</code>
Insert last Node of existing linked list	:	by the function, <code>add_lastnode()</code>
Add node In the end of specific Number	:	by the function, <code>addspecificlast_node ()</code>
Add node In the previous of specific Number	:	by the function , <code>addspecificprevious_node()</code>
Delete the node from linked list	:	by the function, <code>delete_node()</code>
Sorting the Inserted value of linked list	:	by the function , <code>sorting_value()</code>
Display the value of Linked list	:	by the function, <code>display_value()</code>

## REFERENCES

- [1]C: THE COMPLETE REFERENCE (4TH ED) , AUTHOR : HERBERT SCHILDT.
- [2] C Programming Language (2nd Edition By B. W. Kernighan & D. M. Ritchie)
- [3] C Programming: A Modern Approach, 2nd Edition By K. N. King
- [4] Data Structures Using C - Aaron M. Tenenbaum
- [5]Prof. (Dr.) PranamPaul,“An Application to ensure Security through Bit-level Encryption”, International Journal of Computer Science and Network Security, Vol. 9, No. 11, 2009.
- [6]C Programming in 12 Easy Lessons by Greg Perry
- [7]C: A Reference Manual by Samuel P. Harbison and Guy R. Steele
- [8]C Programming: A Modern Approach by K. N. King
- [9]Learn C The Hard Way by Zed Shaw
- [10]Pranam Paul, SaurabhDutta, A K Bhattacharjee,“AnApproach to ensure Security through Bit-level Encryption withPossible Lossless Compression”, International Journal ofComputer Science and Network Security”, Vol. 08, No. 2, pp.291 – 299,2008
- [11]The C Book by Mike Banahan, Declan Brady and Mark Doran
- [12] The Standard C Library by P.J. Plauger
- [13]Algorithms in C by Robert Sedgewick
- [14]Pointers on C by Kenneth Reek
- [15]John C. Bowman,“Math 422 Coding Theory & Cryptography”, University of Alberta, Edmonton, Canada
- [16][https://en.wikipedia.org/wiki/C\\_%28programming\\_language%29](https://en.wikipedia.org/wiki/C_%28programming_language%29)
- [17] <http://www.c4learn.com/data-structure/c-program-to-create-singly-linked-list/>
- [18]<http://piyushgolani.blogspot.in/2012/09/why-use-double-pointer-when-adding-node.html>
- [19]Prof. (Dr.) PranamPaul,“Implementation of InformationSecurity based on Common Division”, International Journal of Computer Science and Network Security, Vol.11, No. 2,2011.
- [20]<https://www.daniweb.com/programming/software-development/threads/377857/creating-a-linked-list-using-function-outside-of-main>

## BIOGRAPHY



Sudip Sinha is a Final year Student of MCA in Narula Institute of Technoloy , Agarpara, WestBengal India.

He completed his Bsc degree from Bhairab Ganguly College (kolkata) under the WBSU .



## International Journal of Innovative Research in Computer and Communication Engineering

*(An ISO 3297: 2007 Certified Organization)*

**Vol. 4, Issue 5, May 2016**



Pritam Ghosh is a Final year Student of MCA in Narula Institute of Technology , Agarpara, WestBengal India. He completed his BCA degree from George College (Kolkata) under the WBUT.



Prof. Dr Pranam Paul, Assistant Professor and Departmental Head, CA Department, Narula Institute of Technology (NIT), Agarpara had completed MCA in 2005. Then my carrier had been started as an academican from MCKV Institute of Technology, Liluah. Parallely, at the same time, I continued my research work. At October, 2006, National Institute of Technology (NIT), Durgapur had agreed to enroll my name as a registered Ph.D. scholar. Then I had joined Bengal College of Engineering and Technology, Durgapur. After that Dr. B. C. Roy Engineering College hired me in the MCA department at 2007. At the age of 30, I had got Ph.D. from National Institute of Technology, Durgapur, WestBengal. I had submitted his Ph.D. thesis only within 2 Years and 5 Months. After completing the Ph.D., I had joined Narula Institute of

Technology in Computer Application Department. Parallely I continue my research work. For that, I have 39 International Journal Publications among 54 accepted papers in different areas. I also reviewer of International Journal of Network Security (IJNS), Taiwan and International Journal of Computer Science Issue (IJCSI); Republic of Mauritius.

Achievements:

Accepted my name for publication in “Who’s Who Science and Engineering, 2011 – 2012” published by “Marquis Who’s Who”, USA on 31st Dec 2010

1. Selected his name as “Top 100 Engineers’ 2011”, by “International Biographical Centre”, Cambridge, England
2. Selected his name as “Outstanding 2000 Intellectuals of the 21st Century, 2012”, by “International Biographical Centre”, Cambridge, England.