



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

Appending Stack and Queue to Enhancement of C Compiler

Pritam Ghosh¹, Sudip Sinha¹, Prof. (Dr.) Pranam Paul²

MCA Final Year Student, Narula Institute of Technology, Agarpara, Kolkata, West Bengal, India¹

HOD, Department of Computer Application, Narula Institute of Technology, Agarpara, Kolkata, West Bengal, India²

ABSTRACT: In this paper, we try to add some user define functions in C compiler like printf() , scanf(). Different operations of Stack and Queue, which are not present in C – Compiler, are being tried add as library functions. Then to operate different kinds of stack and queue 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: Stack, Queue, top, rear, front, C- Compiler, Library File, Header File, Byte Code, Object File, Linker, Loader.

I. INTRODUCTION

Stack is an abstract data type that serves as a collection of elements, with two principal operations: push, which adds an element to the collection, and pop, which removes the most recently added element that was not yet removed. The order in which elements come off a stack gives rise to its alternative name, LIFO (for last in, first out). A stack is a basic data structure that can be logically thought as linear structure represented by a real physical stack or pile, a structure where insertion and deletion of items takes place at one end called top of the stack.

Queue is simply a linear list of information that is associated in first-in , first-out order, which is sometimes called FIFO. That is , the first item placed on the queue is the first item retrieved, the second item put in is the second item retrieved , and so on. This is the only means in storage and retrieval in a queue; random access of any specific item is not allowed. To store information in such way that can be accessible from the beginning element. Like FIFO. It mainly used reservation system and banking system.

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 with maintains Stack and Queue concept. In that case, implementing the stack or queue has been coded, but it may be that the operation of stack or queue is supporting program for helping or calculating the other things. For example if in any program, BFS or DFS is needed, before any thing, to implementing BFS or DFS, stack or queue which is not a used directly here, has to be coded as C – compiler can not provide stack or queue as in build one. If we can include the all operations of stack and queue as in build operations. 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.

The basic concept of stack can be illustrated by thinking of our data set as a stack of plates or books where we can only take the top item off the stack in order to remove things from it [13]. This structure is used all throughout programming in normal stack, at run time only one stack is created. But we created a single function for push and pop for stack, where at run time dynamically, pushing and popping of stack is controlled not only on stack, but also on multiple stacks, defined according to the requirements. Same thing is being done for Queue also. We push a given word to stack - letter by letter - and then pop letters from the stack as well as queue also.



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

III. ACTUAL WORK

2.1 Define the Structure of the stack:

We have implemented the function of stack_push() , stack_pop() and stack_display(), depending a defined structure as follows..

```
struct stack
{
    data_type stk [32766];
    int top;
};
```

2.1.1 PUSH () Function Implemented:

Here we use stack_push() function to enter values multiple times into stack. Here we used 2 arguments named stackToChange as structure pointer and num as a data_type variable in this push() function. Using 1st argument stackToChange to keep the address of stack and 2nd argument is using to insert the value.

```
void stack_push (struct stack*stackToChange, data_type num)
{
    if (stackToChange->top == 32765)
    else
    {
        stackToChange->top = stackToChange->top + 1;
        stackToChange->stk[stackToChange->top] = num;
    }
}
```

2.1.2. POP() Function Implemented :

Here we used stack_pop()function to delete value from stack. Here we used 1st arguments named stackToChange as structure pointer in this pop() function. Argument is used to hold the address of stack.

```
data_type stack_pop (struct stack* stackToChange)
{
    data_type num;
    if (stackToChange->top == - 1)
    {
        return (stackToChange->top);
    }
    else
    {
        num = stackToChange->stk[stackToChange->top];
        stackToChange->top = stackToChange->top - 1;
        return(num);
    }
}
```

2.1.3. Stack Display() Function :

Here we used this display function to display the stack. It is taken only 1 argument for the variable of stack, which is to be display through this function.

```
void stack_display (struct stack stackToChange)
{
    data_type i;
    if (stackToChange->top == -1)
```



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
{
    printf ("Stack is empty\n");
    return;
}
else
{
    for (i = stackToChange->top; i >= 0; i--)
        printf ("%d\n", stackToChange->stk[i]);
}
printf ("\n");
}
```

2.2 Define the Structure of the Queue:

Queue is also an abstract data type or a linear data structure, in which the first element is inserted from one end called REAR, and the deletion of existing element takes place from the other end called as FRONT. This makes queue as FIFO data structure, which means that element inserted first will also be removed first. To implement this, a structure has been defined.

```
struct QUEUE
{
    data_type q [32765];
    data_type front, rear;
};
```

2.2.1. Queue Push() Function Implemented :

Here we used queue_push() function to enter value in the queue. Here we used pointer q as a data type variable which is declared into structure. 1st argument holds the value of queue. 2nd argument is using insert the value.

```
void queue_push (struct QUEUE *q, data_type x)
{
    if ((q->front==0&&q->rear==32764)|| (q->front>0 && q->rear==q->front-1))
    else
    {
        if (q->rear==32764&&q->front>0)
        {
            q->rear=0;
            q->q[q->rear]=x;
        }
        else
        {
            if ((q->front==0&&q->rear== -1)|| (q->rear!=q->front-1))
            {
                q->rear=q->rear+1;
                q->q[q->rear]=x;
            }
        }
    }
}
```

2.2.2 Queue Pop() Function Implemented :

Here we use queue_pop() function to delete value from the queue. Here we used pointer q as a data type variable which is declared into structure.



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
data_type queue_pop (struct QUEUE *q)
{
    data_type a;
    if(q->front==q->rear)
    {
        a=q->q[q->front];
        q->rear=-1;
        q->front=0;
    }
    else
    {
        if(q->front==32766)
        {
            a=q->q[q->front];
            q->front=0;
        }
        else
            a=q->q[q->front++];
    }
    return (a);
}
```

2.2.3. Queue Display() Function Implemented :

Here we used this display function to display the queued. It is taken only 1 argument for the variable of stack, which is to be display through this function.

```
void queue_display(struct QUEUE *q)
{
    int i ,j;
    if(q->front==0&&q->rear==-1)
    {
        printf("Queue is underflow\n");
        getch();
        exit(0);
    }

    if(q->front>q->rear)
    {
        for(i=0;i<=q->rear;i++)
            printf("\t%d",q->q[i]);
        for(j=q->front;j<=32465-1;j++)
        {
            printf("\t%d",q->q[j]);
            printf("\nrear is at %d\n",q->q[q->rear]);
            printf("\nfront is at %d\n",q->q[q->front]);
        }
    }
    else
    {
        for(i=q->front;i<=q->rear;i++)
        {
```

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

```
        printf("\t%d",q->q[i]);
    }
    printf("\nrear is at %d\n",q->q[q->rear]);
    printf("\nfront is at %d\n",q->q[q->front]);
}
}
```

2.3 Using command to create .obj file and .lib file :

TCC – We are using tcc command for creating .obj file on the implemented function of the stack and queue. Step 1. At first we write the code of stack and queue function in notepad and save as it .C extension. Step 2. Then go to the dos prompt and write the command tcc -c filename.c Step 3. Create the .obj file for the stack and queue function.

Ex. tcc -c stack.c

Ex. tcc -c queue.c

TLIB- We are using tlib command for creating .lib file of the stack and queue function. Step1. At first we write the only function for the stack_push , stack_pop, stack_display, queue_push, queue_pop, queue_display because lib file contains only function body. Step2. Then go to the dos prompt and write the command tlib /C mylib.lib+filename.obj Step 3. Create the .lib file for the particular function of the stack and queue.

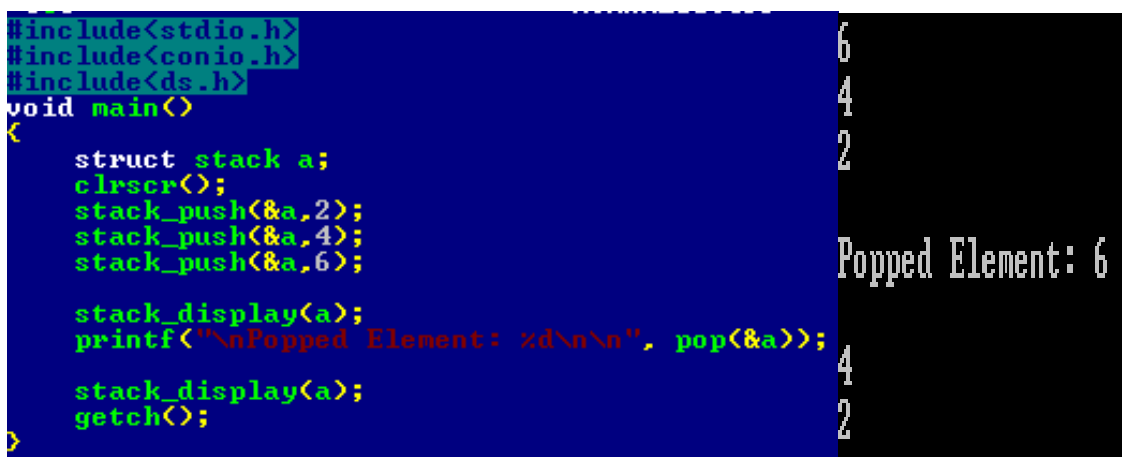
Ex . tlib /C mylib.lib + stack.obj

Ex . tlib /C mylib.lib + queue.obj

Header file – We also create the header file for the stack and queue program. Header file contains only the prototype of the functions that is attached in the main program file before compilation. We write down the only prototype of the function in the notepad and that save into .h extension. In this process we implemented the header file of the particular function. Example: #include<ds.h>

IV. EXAMPLE AND EXECUTION

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 3.1 shows the example of push() and pop () for stack and output display().



```
#include<stdio.h>
#include<conio.h>
#include<ds.h>
void main()
{
    struct stack a;
    clrscr();
    stack_push(&a,2);
    stack_push(&a,4);
    stack_push(&a,6);

    stack_display(a);
    printf("\nPopped Element: %d\n\n", pop(&a));
    stack_display(a);
    getch();
}
```

Figure 3.1

Coding for calling of push() and pop () for stack and output

In Figure 3.2, way of calling functions for push() and pop() for queue and output listis shown with its successful output of execution of program.

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<ds.h>
void main()
{
    struct QUEUE a;
    clrscr();
    queue_push(&a,2);
    queue_push(&a,4);
    queue_push(&a,6);

    queue_display(a);
    printf("\nPopped Element: %d\n\n", pop(&a));

    queue_display(a);
    getch();
}
```

2 4 6
Popped Element: 2
4 2

Figure 3.2
Coding for calling of push() and pop() for queue and output

V. ANALYSIS

We can declare 32767 number of integer variables consecutively but here we have declared 32766 number of integer variables as here top itself is a variable, so top=1 and array = 32765. Here before inserting the data we have incremented the top by 1, if we need to insert the top in 0th position. We can declare this in main () by writing stack.top= -1.

We implemented the stack function, where we have implemented the multiple stacks. We have called push() function to insert values in stack . We have used StackToChange to declare multiple push() function in stack.

We have implemented the single queue by using the circular queue. We can declare 32767 number of integer variables consecutively but here we have declared 32765 number of integer variables as here front itself is a variable, so front=1 and rear itself a variable ,so rear =1 and array = 32764.

In Memory allocation of simple queue, allocate new, larger block of memory reflective of the entire (existing) queue and the queue-size addition. Copy the initial block of memory (i.e. the entire existing queue) to the newly allocated block of memory. Reset the "high-water" variable indicating the new size of the queue. Whereas circular queue, Allocate only the new object or structure-type required. Insert the new allocation anywhere we want in the queue.

The principle advantages in using a circular queue here is no explicit reference to queue indices and the ability to search by queue element data in a way that obfuscates the queue itself. This would be very valuable if your software will be enhanced in the future or part of a larger software design.

VI. CONCLUSION

In this enhanced compiler, it is clearly noded that without writing the coded for following the stack and queue 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 ds.h which is only needed to include for calling the these functions.

1. Insert for push function into the stack : by the function, stack_push()
2. Delete for pop function into the stack : by the function, stack_pop()
3. Show for display function into the stack : by the function, stack_display()
4. Insert for push function into the queue : by the function, queue_push()
5. Delete for pop function into the queue : by the function , queue_pop()
6. Show for display function into the queue : by the function, queue_display()



International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

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
- [17] <http://www.c4learn.com/c-programs/c-program-to-implement-stack-operations>
- [18] [http://www.cprogrammingtutorial.org/cprogram/Write--C-programs-that-implement-stack-](http://www.cprogrammingtutorial.org/cprogram/Write--C-programs-that-implement-stack-operations)
- [19] [https://sites.google.com/site/itstudentjunction/lab-programming-solutions/ data-structures- programs/program-to-implement-queue-using-pointers-in-c](https://sites.google.com/site/itstudentjunction/lab-programming-solutions/data-structures- programs/program-to-implement-queue-using-pointers-in-c)
- [20] [mmingtutorial.org/cprogram/Write-C-programs-that-implement-Queue-\(its-operations\)-using-Pointers/](http://www.cprogrammingtutorial.org/cprogram/Write-C-programs-that-implement-Queue-(its-operations)-using-Pointers/)
- [21] <http://www.geeksforgeeks.org/queue-using-stacks/>

BIOGRAPHY



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.



Sudip Sinha is a Final year Student of MCA in Narula Institute of Technology , Agarpara, WestBengal India. He completed his Bsc degree from Bhairab Ganguly College (kolkata) under the WBSU.



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 academicians 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:



ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 5, May 2016

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.