DOPE (/blog/)

Making a stack using linked list in C

May 26, 2017 C (/blog/tag/c/?tag=c) STACK (/blog/tag/stack/?tag=stack) LINKED LIST (/blog/tag/linked-list/?tag=linked-list) DATA STRUCUTRE (/blog/tag/data-strucutre/?tag=data-strucutre) 30109

Previous:

- Stacks in C (https://www.codesdope.com/blog/article/stacks-in-c/)

The previous article (https://www.codesdope.com/blog/article/stacks-in-c/) was all about introducing you to the concepts of a stack. In this article, we will code up a stack and all its functions using a linked list.

The first thing required to make a stack using a linked list is obviously a linked list. So, let's start by creating a linked list.

```
struct node
{
    int data;
    struct node *next;
};
typedef struct node node;

node *top;
```

The concepts and the codes of a linked list are explained in the article "Linked list in C (https://www.codesdope.com/blog/article/linked-lists-in-c-singly-linked-list/)".

`typedef struct node node` – In this line of code, we are just representing `struct node` with `node` by using *typedef*. You can learn about typedef from the typedef chapter of the C course (https://www.codesdope.com/c-typedef/).

The next thing to create a stack is the creation of the 'top' pointer. This is done in the next line of the code i.e., `node *top`. We have created a node (struct node) using this line of code.

Now, we need to initialize our stack and this will be done by making the 'top' equal to NULL because at this point of time our stack doesn't contain any element.

```
void initialize()
{
    top = NULL;
}
```

This is the initialize function to initialize the stack by making the top NULL.

The next and the most important operations on a stack are ***push*** and ***pop***. So, let's create them.

push

The steps for push operation are:

1. Make a new node.

2. Give the 'data' of the new node its value.

3. Point the ‘next’ of the new node to the top of the stack.

4. Make the ‘top’ pointer point to this new node.

```
void push(int value)
{
    /*
    Let the initial stack is:
       Top
     ___|___         _______         _______
    |    1   |____\ |    3   |____\ |    5   |____\ NULL
    |________|    / |________|    / |________|    /
    and the value to be pushed is 10
    */
    node *tmp;
    tmp = malloc(sizeof(node));
    tmp -> data = value;
    /*
     This will make a new node
     _______
    |   10   |
    |________|
    */
    tmp -> next = top;
    /*
    This step will do
                       Top
     _______         ___|___         _______        _______
    |   10   |____\ |    1   |____\ |    3   |____\| 5     |____\ NULL
    |________|    / |________|    / |________|    /|_______|    /
    */
    top = tmp;
    /*
    This step will do
       Top
     ___|___         _______         _______        _______
    |   10   |____\ |    1   |____\ |    3   |____\| 5     |____\ NULL
    |________|    / |________|    / |________|    /|_______|    /
    */
}
```

Read the comments in the code for better understanding.

The first step is to make a new node and we are doing the same by

`node * tmp`

`tmp  = malloc(sizeof(node))`

The second step is to give ‘data’ of this new node its value and this we are doing with `tmp  -> data = value`.

The third step is to point the 'next' of the new node to the top of the stack and this is done in the next line – `tmp -> next = top.`

The last step is to make the 'top' pointer point to this new node – `top = tmp`

You must have understood the push operation. So, let's deal with the pop operation now.

pop

In pop operation, we delete the topmost node and returns its value. In order to do so, we need to make the 'top' pointer point to the node next to the topmost node but this will led the current topmost node inaccessible. So, we will first make a temporary pointer to the current top node and delete it using the 'free (https://www.codesdope.com/c-dynamic-memory/)' function later. The steps for the pop operations are:

1. Make a temporary node.
2. Point this temporary node to the top of the stack
3. Store the value of 'data' of this temporary node in a variable.
4. Point the 'top' pointer to the node next to the current top node.
5. Delete the temporary node using the 'free (https://www.codesdope.com/c-dynamic-memory/)' function.
6. Return the value stored in step 3.

```
int pop()
{
    node *tmp;
    int n;
    tmp = top;
    n = tmp->data;
    top = top->next;
    free(tmp);
    return n;
}
```

The code is very simple and just follows the steps mentioned above.

`node * tmp` – Step 1

`tmp  = top` – Step 2

`n = tmp->data` – Step 3

`top = top->next` – Step 4

`free(tmp)` – Step 5

`return n` – Step 6

Now, let’s create some functions for the other operations also e.g., top, isempty, etc.

```
int Top()
{
    return top->data;
}

int isempty()
{
    return top==NULL;
}
```

‘isempty’ is to check if the stack is empty or not and this can be done by checking the ‘top’ pointer. If the ‘top’ pointer is null then the stack is empty otherwise not and this is what we are doing in the ‘isempty’ function.

The ‘Top’ function is just returning the data at the top of the stack.

So, the overall code would be:

```c
#include <stdio.h>
#include <stdlib.h>
#define TRUE 1
#define FALSE 0

struct node
{
    int data;
    struct node *next;
};
typedef struct node node;

node *top;

void initialize()
{
    top = NULL;
}

void push(int value)
{
    node *tmp;
    tmp = malloc(sizeof(node));
    tmp -> data = value;
    tmp -> next = top;
    top = tmp;
}

int pop()
{
    node *tmp;
    int n;
    tmp = top;
    n = tmp->data;
    top = top->next;
    free(tmp);
    return n;
}

int Top()
{
    return top->data;
}

int isempty()
{
    return top==NULL;
}

void display(node *head)
{
    if(head == NULL)
    {
        printf("NULL\n");
    }
    else
    {
        printf("%d\n", head -> data);
```

```
            display(head->next);
        }
}

int main()
{
    initialize();
    push(10);
    push(20);
    push(30);
    printf("The top is %d\n",Top());
    pop();
    printf("The top after pop is %d\n",Top());
    display(top);
    return 0;
}
```

You can learn about creating a stack using an array in the next article.

Next:

- Making a stack using an array in C (https://www.codesdope.com/blog/article/making-a-stack-using-an-array-in-c/)