Stack data structure (Stack)

A stack is an abstract data structure (Abstract Data Type - ADT for short), mostly used in almost every programming language. Name the stack because it acts as a stack in real life, such as a deck of cards or a stack of disks ...

What is the stack (Stack)?

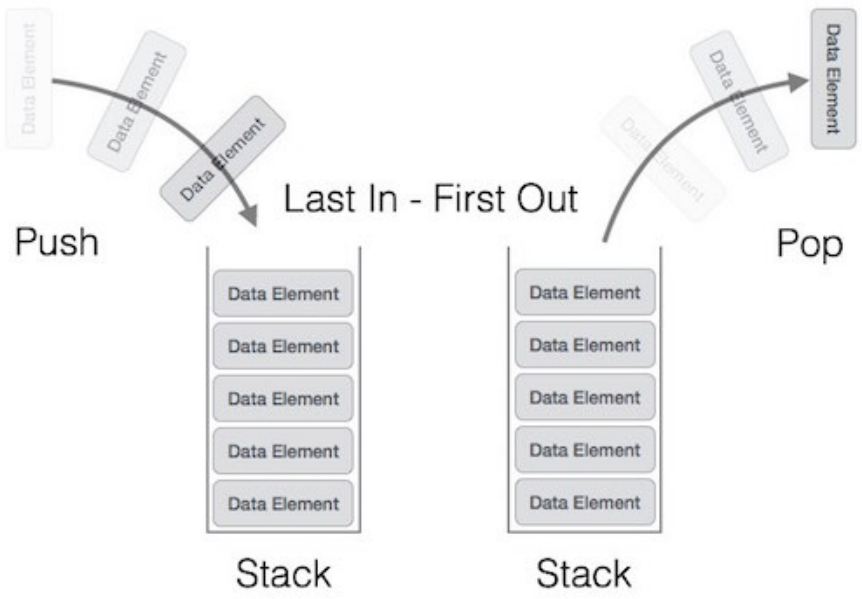
A stack is an abstract data structure (Abstract Data Type - ADT for short), mostly used in almost every programming language. Name the stack because it acts as a stack in real life, such as a deck of cards or a stack of disks .

In real life, the stack only allows operations at the top of the stack. For example, we can only place or add a card or a disk to the top of the stack. Therefore, the abstract data stack structure only allows data manipulation at the top position. At any time, we can only access the top element of the stack.

This feature makes the stack become LIFO format data structure. LIFO stands for Last-In-First-Out. Here, the element placed (inserted, added) will eventually be accessed first. In stack terminology, insert operations are called **PUSH** operations and delete operations are called **POP** operations.

Demonstration of stack data structure (Stack)

Below is a diagram illustrating a stack and the activities taking place on the stack.



A stack can be deployed by Array method (Array), Structure (Struct), Cursor (Pointer) and Linked List (Linked List). The stack may be in a fixed size or a resizable stack. Below we will deploy the stack using arrays with deployment of fixed stacks.

Basic operations on the stack data structure

The basic operations on the stack may involve initializing the stack, using it and then deleting it. In addition to these basic operations, a stack has two primitive activities related to the concept, namely:

Push operation () : store an element on the stack.

Pop () operation : delete an element from the stack.

When the data has been PUSH up on the stack:

In order to use the stack effectively, we also need to check the status of the stack. For this purpose, here are some other support features of the stack:

Peek () operation : get the data element at the top of the stack, without deleting this element.

IsFull () operation : check if the stack is full.

Operation isEmpty () : check if the stack is empty or not.

At all times, we maintain a pointer to the last PUSH data element on the stack. Because this pointer always represents the top position of the stack so is named **top. The top pointer** gives us the value of the top element of the stack without having to perform the above delete operation (pop operation).

In the next section, we will learn about methods to support stack features.

The peek () method of the stack data structure

The peek () function algorithm:

```
B ? t  ?? u h à m peek  return  stack [ top ]  k ? t th ú c h à m
```

Deployment of the peek () function in C language:

```
int  peek () { return  stack [ top ]; }
```

The isFull () method of the stack data structure

The algorithm of isFull () function:

```
B ? t  ?? u h à m isfull  if top  b ?
ng MAXSIZE  return true else return false  k ? t th ú c  if  k ?
t th ú c h à m
```

Deployment of isFull () function in C language:

```
bool  isfull () { if ( top  ==  MAXSIZE ) return true ; else return false ; }
```

The isEmpty () method of the stack data structure

The algorithm of isEmpty () function:

```
b ? t  ?? u h à m isempty  if top  nh ?  h ?
n  1 return true else return false  k ? t th ú c  if  k ? t th ú c h à m
```

The implementation of isEmpty () function in C language is a bit different. We initialize the top at -1, just like the index of the array starts from 0. So we check if the top is below 0 or -1, the stack is empty. Here is the code:

```
bool  isempty () { if ( top  == - 1 ) return true ; else return false ; }
```

PUSH operation in the stack data structure

The process of placing (adding) a new data element on the stack is also known as PUSH Activity. Push activities include the following steps:

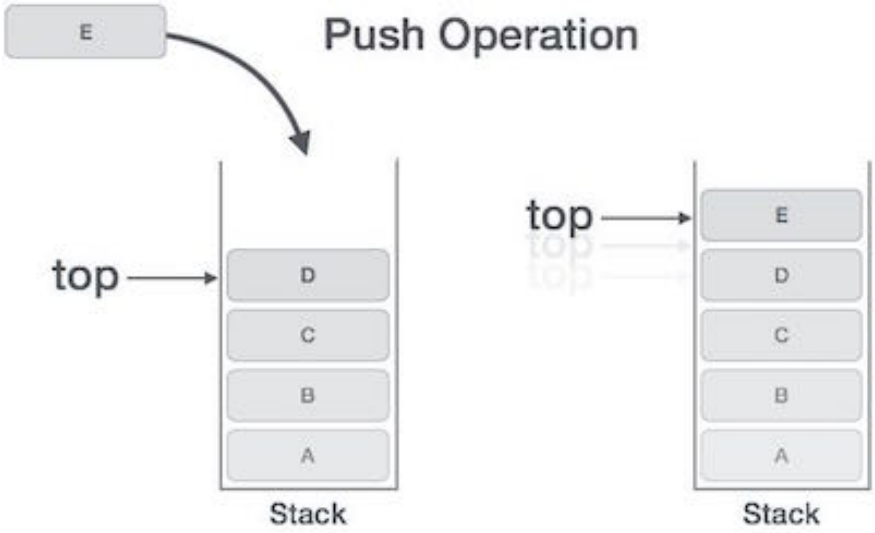
Step 1 : check if the stack is full.

Step 2 : If the stack is full, the process fails and exit.

Step 3 : If the stack is not full, increase the top to point to the next available memory.

Step 4 : Add the data element to the location where the top is pointing to the stack.

Step 5 : return success.



If the Linked List is used to deploy the stack, then in step 3 we need to allocate a dynamic space.

Algorithm for PUSH operation of stack data structure

The above word can deduce a simple algorithm for PUSH operation in the stack data structure as follows:

```
b ? t  ?? u ho ? t  ?? ng push :  stack ,  data  if  stack l à ??
y  return null  k ? t th ú c  if  top  ?  top  + 1  stack [ top ] ?
 data k ? t th ú c h à m
```

The implementation of this algorithm in C language is:

```
void  push ( int  data ) { if (! isFull ()) {  top  =  top  + 1 ;  stack [ top
```

To learn about the C program that fully illustrates the above activities of the stack, please go to chapter: Stack (Stack) in C.

POP operation of the stack data structure

Accessing element content while deleting it from the stack is also known as POP Activity. In the Array deployment of pop () operation, the data element is not actually deleted, instead the top is reduced to a lower position in the stack to point to the next value. But in the Linked List implementation, the pop () operation actually deletes the data element and deletes it from the memory space.

POP operations may include the following steps:

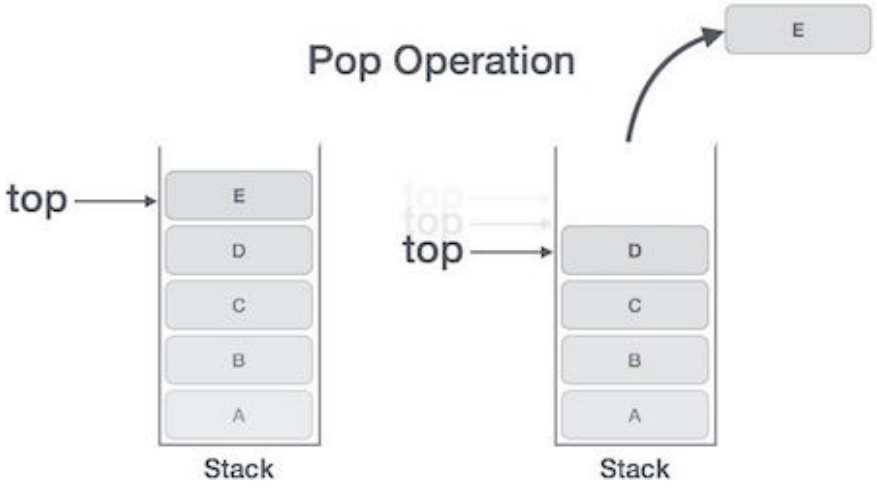
Step 1 : check if the stack is empty or not.

Step 2 : If the stack is empty, the process fails and exit.

Step 3 : If the stack is not empty, accessing the data element at the top is pointing to.

Step 4 : reduce the value of the top 1.

Step 5 : return success.



Algorithms for POP operations

From the above we can deduce the algorithm for POP operation on the stack data structure as follows:

```
b ? t  ?? u h à m pop :  stack  if  stack l à  tr ? ng  return null  k ?
t th ú c  if  data  ?  stack [ top ]  top  ?  top  - 1 return  data k ?
t th ú c h à m
```

The implementation of algorithms in C language is as follows:

```
int  pop ( int  data ) { if (! isempty ()) {  data  =  stack [ top ];  top  =
```

To learn the C program fully illustrates the above activities of the stack, invite you to the chapter: **Stack (Stack) in C.**

Application of stack

- Handling function calls in C / C ++ - In computers, used to calculate expression values, interrupt handling - In compilation programs - In web browsers, text editors - Evaluating expressions + Neutral expression: the two-operator operator stands between two operands, the one-operator operator precedes the operand + The suffix expression: operator after the operand + Prefix expression: the operator precedes the operand

Example valuation of expression A = b + c * d / e - f Element a * (bc) / d Abc suffix- * d / Prefix / * a-bcd

Browse suffix expressions:

- Meet the operand: push on the stack - Meet the 1-star operator: get 1 operand in the stack, apply the operator to the operand and then return the stack - Meet the 2-operator operator: get 2 operands at the top stack in sequence, apply the operator to the two operands, the result is pushed back to the stack - Finish, give the result that the value at the top of the stack - Vd evaluate the suffix expression

Convert the intermediate expression to the suffix

Browse in turn the central symbol from left to right - Meet operands: write to the result expression - Meet the operator with priority less than 6 + If the stack is empty or the stack is an operator with a lower priority or is '(' push the operator into consideration of the stack + On the contrary: get the operators at the top of the stack with a higher priority or with the operator taking turns putting into the final result and pushing the operator into the stack. - Meet the operator with priority 6 or '(' then push the stack - Meet ')' get all the operators in the stack until the '(' first, put the result expression in the correct order and push a symbol '(' out of the stack - If all the intermediate expressions are removed, take the operators in the stack to put the result expression in the correct order.

According to Tutorialspoint

Previous article: Loop List List (Circular Linked List)

Next lesson: Queue data structure (Queue)