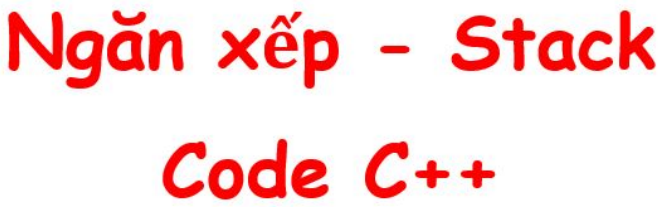
Ngăn xếp Stack – Cài đặt ngăn xếp bằng mảng code C/C++

By **Như Quỳnh** - 29/05/2020



Ngăn xếp và hàng đợi là những **cấu trúc dữ liệu** quan trọng. Bài viết chia sẻ **code ngăn xếp stack C/C++** cài đặt bằng mảng một chiều con trỏ. Cách ứng dụng của stack vào bài tập quản lý danh sách liên kết.

Mục lục bài viết

1. Ngăn xếp – Stack là gì?

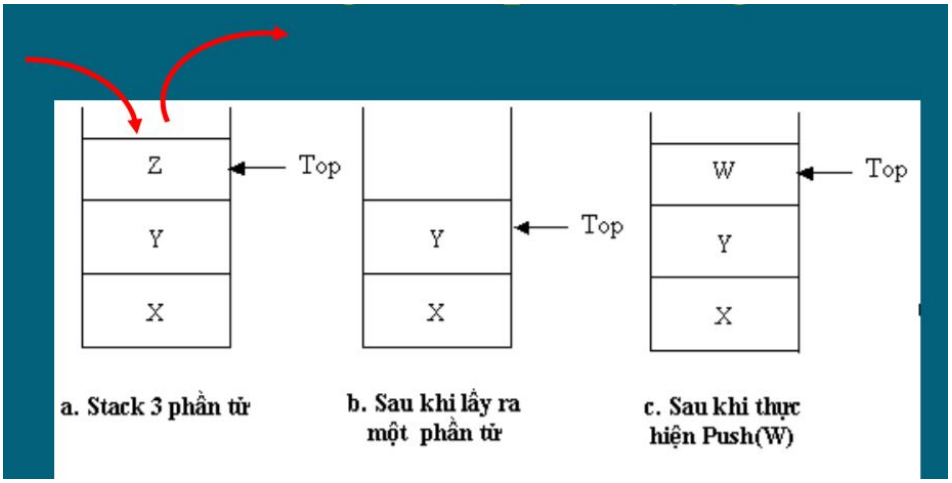
Ngăn xếp (**Stack**) **trong C ++** là một cấu trúc dữ liệu tương tự như danh sách liên kết đơn, hay Queue hàng đợi.
dạng danh sách liên kết đặc biệt trong đó việc thêm vào, lấy ra phần tử chỉ thực hiện ở một đầu của danh sách.

Đặc điểm của ngăn xếp: Tuân theo quy luật **" vào trước ra sau "** hay là **" vào sau ra trước"** (FILO – First In Last Out hoặc LIFO – Last In First OUT)

Bạn hình dung hoạt động của cấu trúc này tương tự như việc bạn xếp một chồng bát, đĩa. Cách bạn thêm vào và lấy ra chúng cho dễ hiểu.

Ngoài ngôn ngữ lập trình C/C++, thì tất cả các ngôn ngữ khác như Python, C#, PHP, Java, Javascript . . . đều sử dụng tới cấu trúc này!

Hình ảnh mô phỏng ngăn xếp:



1.1 Các dạng biểu diễn của ngăn xếp

Thực chất **stack** là một **danh sách liên kết** , nên bạn có thể sử dụng một trong các dạng cài đặt sau:

1. Cài đặt stack bằng danh sách liên kết
2. Biểu diễn trực tiếp ngăn xếp bằng mảng một chiều
3. Cài đặt bằng con trỏ (sử dụng các node stack)

Ở bài viết này mình sẽ sử dụng cách thứ 2 tức sử dụng mảng 1 chiều.
Xem ngăn xếp stack cái đặt bằng con trỏ.

1.2 Các phép toán trên ngăn xếp

Đối với cấu trúc dữ liệu stack, bạn chắc chắn sẽ phải sử dụng những phép toán sau:

- Tạo ngăn xếp rỗng : Hàm Init (S)
- Thêm phần tử vào đầu ngăn xếp : Hàm Push(S,x)
- Lấy phần tử khỏi danh sách: Hàm Pop(S)
- Kiểm tra ngăn xếp rỗng: Hàm Isempty(S)
- Chèn phần tử vào vị trí bất kì: Hàm Insert_k (S, x, k)

Lưu ý rằng các tên hàm mình kể bên trên là do bạn tự định nghĩa. Vì code trong bài mình sẽ đặt tên hàm như vậy nên mình viết cho bạn đọc dễ hiểu thôi nhé!

2. Ngăn xếp cài đặt bằng mảng

Bài toán: Cài đặt **ngăn xếp stack** các số nguyên, viết hàm chèn phần tử vào vị trí bất kỳ!

Mình sẽ **sử dụng ngôn ngữ C++** để giải quyết bài toán nhé! Nếu như bạn đang tìm code C thì chỉ cần thay đổi câu lệnh nhập xuất **cin, cout** thành **scanf, printf** là được.
Hoặc xem phần 4 có nha!

> **Chú ý:** Lỗi hiển thị, mình không biết tại sao nhưng tất cả dấu **&** trong bài viết đều chuyển thành &**amp ;**

```
21  //Khoi tao ngan xep
22  void Init (Stack & S){
23          S.Top = 0;
24  }
```

2.1 Khai báo cài đặt ngăn xếp

Ngăn xếp stack cài đặt bằng mảng một chiều sẽ có một biến Top dùng để lưu vị trí hiện tại, vị trí đỉnh của ngăn xếp.
Thành phần thứ 2 là mảng chứa Data dữ liệu.

```
1  //
2  // Cai dat ngan xep
3  typedef int item;    // Kieu cua ngan xep
4  #define Max 100      // So phan tu toi da cua Stack
5  struct Stack{
6      int Top;
7      item Data[Max];
8  };
9  Stack S;  // Khai bao ngan xep S
```

2.2 Nhóm hàm khởi tạo và kiểm tra

Hàm khởi tạo ngăn xếp: Chỉ cần cho giá trị của biến Top = 0 là xong.
Hàm kiểm tra rỗng thì ngược lại trả về đúng nếu biến Top == 0 là được.

```
01  //Khoi tao ngan xep
02  void Init (Stack & S){
03      S.Top = 0;
04  }
05
06  //Kiem tra ngan xep rong
07  int Isempty( Stack S){
08      return (S.Top==0);
09  }
10
11  //Kiem tra ngan xem day
12  int Isfull(Stack S){
13      return (S.Top == Max);
```

```
}
```

2.3 Thêm phần tử vào đầu stack – Hàm Push

Hàm Push dùng để thêm phần tử vào đầu ngăn xếp.
Nếu ngăn xếp đã đầy thì không cho phép thêm, còn ngược lại thì:

Tăng biến Top lên một đơn vị rồi gán Data ở vị trí Top bằng x. Như vậy giá trị x đã được thêm vào đầu danh sách.

```
//
//Ham Push them phan tu x vao dau ngan xep
void Push(Stack & S, item x){
    if(Isfull(S))
        cout<<"\nNgan xep day!"<<endl;
    else{
        S.Top ++;
        S.Data[S.Top]=x;
    }
}
```

2.4 Hàm lấy phần tử đầu – Pop Stack

Hàm này bản chất tương tự hàm Push. Bạn cần khai báo một biến dùng để đựng giá trị của phần tử đầu stack. Sau đó giảm giá trị của biến Top, return về biến đã lưu.
Đoạn kiểm tra ngăn xếp rỗng có làm hay không là tùy bạn nhé!

```
// Ham Pop lay phan tu khoi dau ngan xep
item Pop(Stack & S){
    if(Isempty(S))
        cout<<"\n Ngan xep rong! "<<endl;
    else{
        item x = S.Data[S.Top];
        S.Top--;
        return x;
    }
}
```

2.5 Hàm chèn vào vị trí k bất kì trong ngăn xếp

Chèn vào vị trí bất kì sẽ phức tạp hơn rất nhiều hàm chèn đầu. Chính vì phải thao tác với stack theo đúng nguyên tắc "Vào trước ra sau " (Fisrt In Last Out) nên mới phức tạp.

Minh sử dụng cả hàm Push và hàm Pop trong hàm này. Ý tưởng đưa ra là khai báo một Stack tạm thời dùng đế lưu trữ giá trị từ vị trí k đến vị trí Pop của danh sách cần chèn.

Tức là bạn lấy giá trị ra khỏi ngăn xếp cho tới vị trí cần chèn , sau đó chèn phần tử cần chèn vào. Sau khi chèn thì lại thêm lại các phần tử vừa lấy ra.

Xem code cho nhanh hiểu nào:

```
//
// Chen phan tu x vao vi tri k
void Insert_k(Stack & S, item x, int k){
    if(k<1 || k>S.Top)
        cout<<"\nVi tri chen khong hop le! "<<endl;
    else{
```

```
        Stack Tempt; // Khai bao stack tam thoi
        Init(Tempt);  // Khoi tao stack

        while(S.Top>=k){  // Chuyen phan tu tu S sang Tempt
            Push(Tempt, Pop(S));
        }

        Push(S,x); // Them phan tu x vao vi tri k

        while(Tempt.Top>0)    // Lay lai gia tri ve S
            Push(S,Pop(Tempt));
    }
}
```

2.6 Hàm nhập phần tử – Input

Nhập như thế nào là tùy theo từng bài toán. Tùy theo cách bạn muốn nhập phần tử vào ngăn xếp.

Ở đây mình quản lý ngăn xếp các số nguyên nên có thế khác vấn đề bạn cần tìm một chút. Tuy nhiên bạn chỉ cần sửa một chút phần nhập dữ liệu từ bàn phím là ok.

Mình viết hàm nếu nhập giá trị là 0 thì sẽ ngừng nhập. Bạn tham khảo nhé!

```
//
// Ham Nhap
void Input(Stack&S){
    cout<<"\nNhap gia tri cho Stack \nNhap 0 de ket thuc "<<endl;
    item x;
    do{
        cin>>x;
        if(x!=0)
            Push(S,x);
    }
    while(x!=0);
}
```

2.7 Hàm xuất – Output Stack

Xuất dữ liệu ra màn hình thì thật đơn giản. Dùng Pop để lấy giá trị và in ra màn hình là ok rồi.

```
//
//Ham xuat
void Output(Stack  S){
    while(S.Top!=0)
        cout<<"  "<<Pop(S);
}
```

3. Chương trình hoàn chỉnh

> Tham khảo toàn bộ bài code này trên Github của mình **tại đây** !
>
> Xem code cài đặt bằng con trỏ **tại đây**

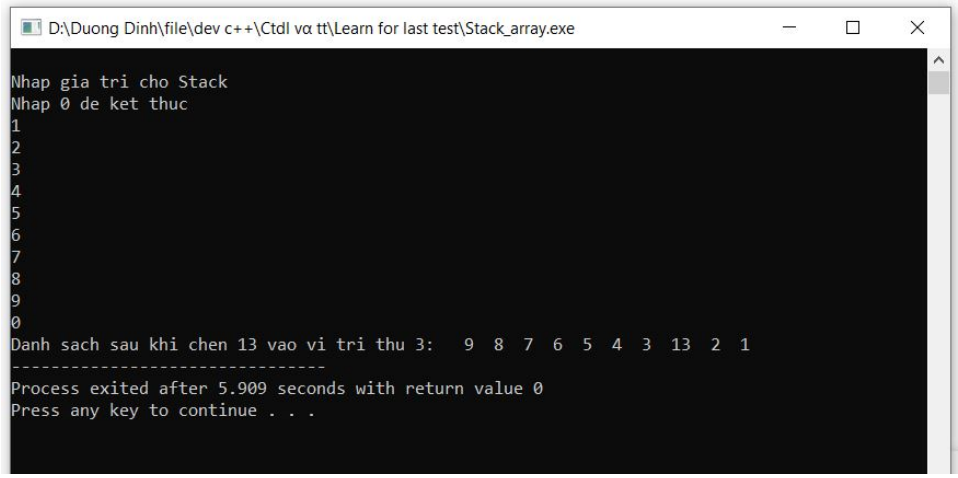
Nếu bạn viết với ngôn ngữ C thì đổi chút khai báo thư viện, lệnh nhập xuất (cin, cout thành scantf, printf tương ứng là được nhé)

Code Stack C++:

```
// Ngan xep cai dat bang mang
/*
Cac ham can viet:
 + Them phan tu vao dinh Push
 + Xoa phan tu khoi dinh Pop
 + Nhap xuat du lieu
 + Them xoa o day phan tu
*/
#include<bits/stdc++.h>
using namespace std;

// Cai dat ngan xep
typedef int item;    // Kieu cua ngan xep
#define Max 100      // So phan tu toi da cua Stack
struct Stack{
    int Top;
    item Data[Max];
};
Stack S;  // Khai bao ngan xep S

//Khoi tao ngan xep
void Init (Stack & S){
    S.Top = 0;
}

//Kiem tra ngan xep rong
int Isempty( Stack S){
    return (S.Top==0);
}

//Kiem tra ngan xem day
int Isfull(Stack S){
    return (S.Top == Max);
}

//Ham Push them phan tu x vao dau ngan xep
void Push(Stack & S, item x){
    if(Isfull(S))
        cout<<"\nNgan xep day!"<<endl;
    else{
        S.Top ++;
        S.Data[S.Top]=x;
    }
}

// Ham Pop lay phan tu khoi dau ngan xep
item Pop(Stack & S){
    if(Isempty(S))
        cout<<"\n Ngan xep rong! "<<endl;
    else{
        item x = S.Data[S.Top];
        S.Top--;
        return x;
    }
}

// Chen phan tu x vao vi tri k
void Insert_k(Stack & S, item x, int k){
    if(k<1 || k>S.Top)
        cout<<"\nVi tri chen khong hop le! "<<endl;
    else{
        Stack Tempt; // Khai bao stack tam thoi
        Init(Tempt);
```

```
64
65          while(S.Top>=k){  // Chuyen phan tu tu S sang Tempt
66              Push(Tempt, Pop(S));
67          }
68
69          Push(S,x); // Them phan tu x vao vi tri k
70
71          while(Tempt.Top>0)    // Lay lai gia tri ve S
72              Push(S,Pop(Tempt));
73      }
74  }
75  // Ham Nhap
76  void Input(Stack&S){
77      cout<<"\nNhap gia tri cho Stack \nNhap 0 de ket thuc "<<endl;
78      item x;
79      do{
80          cin>>x;
81          if(x!=0)
82              Push(S,x);
83      }
84      while(x!=0);
85  }
86
87  //Ham xuat
88  void Output(Stack  S){
89      while(S.Top!=0)
90          cout<<"  "<<Pop(S);
91  }
92
93  int main(){
94      Input(S);
95      Insert_k(S,13,3); // Chen phan tu 13 vao vi tri 3
96      cout<<"Danh sach sau khi chen 13 vao vi tri thu 3: ";
97      Output(S);
98      return 0;
99  }
```

Kết quả khi chạy chương trình trên



Xem cài đặt danh sách liên kết đơn

Xem Quere trong C++

Như Quỳnh