Cấu trúc dữ liệu ngăn xếp stack trong c/c++

Cấu trúc ngăn xếp stack nguyên lý hoạt động LIFO (Last in, first out) vào sau ra trước.

Trong cấu trúc dữ liệu ngăn xếp stack có 2 thao tác chính:

+ Thao tác **push**

+ Thao tác **pop**

A. Cấu trúc ngăn xếp stack với mảng

```
#define MAX 100 // kích thước tối đa là N phần tử ví dụ N=100
struct stack{
        // khai báo biến int top để cho biết chỉ số của đầu stack
        int top;
        // Mảng 1 chiều A[MAX] chứa kiểu dữ liệu số nguyên
        int A[MAX];
};
typedef struct stack STACK;
```

Các thao tác cơ bản khi cài đặt stack bằng mảng bao gồm:

- Khởi tạo stack: Init (S)

```
void Init (STACK &s){
        //gan phan tu top cua stack bang -1 de khoi tao để đảm bảo đúng yêu cầu stack rỗng khi được khởi tạo
        s.top=-1;
}
```

- Kiểm tra stack rỗng: Empty (S)

```
int Empty (STACK s){
        //neu phan tu top bang -1 thi stack rong va return 1
        if (s.top == -1){
```

```
            return 1;
        }
        //nguoc lai return 0
        return 0;
}
```

- Kiểm tra stack đầy: IsFull (S)

```
int IsFull (STACK s){
        //neu phan tu top cua stack bang MAX - 1 thi stack da day thi return 1
        if (s.top==MAX-1){
            return 1;
        }
        //nguoc lai return 0
        return 0;
}
```

- Đưa phần tử vào Stack: Push(S,x)

```
void Push(STACK &s,int x){ // them phan tu vao cuoi mang
        // neu stack chua day thi thuc hien push
        if(IsFull(s)==0){
            //tang chi so top len 1 don vi
            s.top++;
            //gan phan tu mang tai vi tri top bang gia tri x
            s.A[s.top]=x;
        }
}
```

- Lấy phần tử ra khỏi Stack: Pop (S, x)

```
int Pop(STACK &s){ //Lay phan tu o tren dinh top ra va sau do thuc hien xoa luon phan tu do
```

```
        //khai bao bien x de chua du lieu
        int x;
        // neu stack khac rong
        if(!Empty(s)){
                //thuc hien gan phan tu cuoi cua stack bang x
                x = s.A[s.top];
                //giam so luong phan tu mang di 1 don vi
                s.top--;
        }
        //tra ve x duoc gan bang phan tu cuoi cung cua stack
        return x;
}
```

- Hàm top trong stack: Top (S,x)

```
int Top(STACK &s){// Lay phan tu tren dinh top ra va hoan toan khong xoa phan tu của stack
        //khai bao bien x de chua du lieu
        int x;
        // stack khac rong
        if(!Empty(s)){
                //thuc hien gan phan tu cuoi bang x
                x = s.A[s.top];
                //tra ve x duoc gan bang phan tu cuoi cung cua mang
                return x;
        }else{ //neu stack do rong thi tra ve NULL
                return NULL;
        }
}
```

Nhập xuất phần tử vào mảng ngăn xếp stack: sử dụng lại các hàm **push, pop và top**

Hàm nhập N phần tử vào stack như sau:

```
void Input(STACK &s, int n){
	//duyet tu 0 den n
	for(int i = 0; i< n; i++){
		//thuc hien nhap gia tri vao bien x
		int x;
		cout << "Nhap gia tri phan tu thu: " << i << endl;
		cin >> x;

		//thuc hien push vao stack
		Push(s,x);
	}
}
```

Hàm xuất các phần tử trong stack ra như sau:

```
void Output(STACK s){
	//duyet tu phan tu top ve phan tu cuoi stack
	for(int i = s.top; i > -1; i--){
		//hien thi ra ket qua
		cout  << s.A[i] <<endl;

	}
}
```

Chương trình cài đặt stack bằng mảng hoàn chỉnh

```
#include <iostream>
using namespace std;
#define MAX 100
struct stack{
	int top;
	int A[MAX];
};
typedef struct stack STACK;
void Init (STACK &s){
```

```
	//gan phan tu top cua stack bang -1 de khoi tao
	s.top=-1;
}
int Empty (STACK s){
	//neu phan tu top bang -1 thi stack rong va return 1
	if (s.top == -1){
		return 1;
	}
	//nguoc lai return 0
	return 0;
}
int IsFull (STACK s){
	//neu phan tu top cua stack bang MAX - 1 thi stack da day thi return 1
	if (s.top==MAX-1){
		return 1;
	}
	//nguoc lai return 0
	return 0;
}
void Push(STACK &s,int x){
	// neu stack chua day thi thuc hien push
	if(IsFull(s)==0){
		//tang chi so top len 1 don vi
		s.top++;
		//gan phan tu mang tai vi tri top bang gia tri x
		s.A[s.top]=x;
	}
}
int Pop(STACK &s){
	//khai bao bien x de chua du lieu
	int x;
	// neu stack khac rong
	if(!Empty(s)){
```

```
            //thuc hien gan phan tu cuoi cua stack bang x
            x = s.A[s.top];
            //giam so luong phan tu mang di 1 don vi
            s.top--;
    }
    //tra ve x duoc gan bang phan tu cuoi cung cua stack
    return x;
}
int Top(STACK &s){
    //khai bao bien x de chua du lieu
    int x;
    // stack khac rong
    if(!Empty(s)){
            //thuc hien gan phan tu cuoi bang x
            x = s.A[s.top];
            //tra ve x duoc gan bang phan tu cuoi cung cua mang
            return x;
    }else{ //neu stack do rong thi tra ve NULL
            return NULL;
    }
}
void Input(STACK &s, int n){
    //duyet tu 0 den n
    for(int i = 0; i< n; i++){
            //thuc hien nhap gia tri vao bien x
            int x;
            cout << "Nhap gia tri phan tu thu: " << i << endl;
            cin >> x;
            //thuc hien push vao stack
            Push(s,x);
    }
}
void Output(STACK s){
```

```
    //duyet tu phan tu top ve phan tu cuoi trong stack
    for(int i = s.top; i > -1; i--){
            //hien thi ra ket qua
            cout << s.A[i];
    }
}
int main(){
    //tao mot stack s
    STACK s;
    //nhap n phan tu can them vao stac
    int n;
    cout << "Nhap n: ";
    cin >> n;
    //khoi tao stack s
    Init(s);
    //goi ham nhap n phan tu vao stack
    Input(s,n);
    //goi ham xuat cac phan tu o tron stact
    cout << "Stack vua nhap la: \n";
    Output(s);
    //lay phan tu top cua stack ra nhung khong xoa
    cout << "Top cua stack voi ham top" << Top(s)<< endl;
    //lay phan tu top cua stack ra va xoa
    cout << "\nTop cua stack voi ham pop \n" << Pop(s)<< endl;
    cout << "Stack sau khi pop la\n";
    Output(s);
}
```

B. Cấu trúc stack với danh sách liên kết

Thì các phần tử cũng chính là một node trong đó sẽ có 2 thành phần:

+ data

+ con trỏ next

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

Vì stack chỉ thao tác với phần tử ở trên đỉnh của nó (hay còn gọi là phần tử Top) nên việc khai báo stack với danh sách liên kết chỉ đơn giản là việc khai báo một kiểu dữ liệu stack với một NODE *top;

```
struct stack{
NODE *top;
};
typedef struct stack STACK;
```

Các thao tác cơ bản khi cài đặt stack bằng danh sách liên kết bao gồm:

- Khởi tạo stack: Init (S)

```
void Init (STACK &s){
        //gan top cua stack ve NULL va khien cho stack do tro nen rong
        s.top = NULL;
}
```

- Tạo mới 1 nút: CreateNode (x)

  Khi cài đặt stack bằng danh sách liên kết, chắc chắn rằng các phần tử của stack đó sẽ là một node. Và một node thì lại có 2 thành phần đó là **thành phần data** và **thành phần địa chỉ** đến node khác. Vì vậy ta cần hàm khởi tạo node để khai báo đủ 2 thành phần trên trước khi đưa node đó vào stack.

```
NODE* CreateNode (int x) {
	NODE *p;
	p = new NODE;
	//neu p = NULL thi ko du bo nho cap phat
	if (p==NULL) {
		printf ("KHONG DU BO NHO!");
		return NULL;
	}
	//gan data bang X
	p->data=x;
	//gan con tro next bang NULL
	p->next=NULL;
	//tra ve node p
	return p;
}
```

- Kiểm tra stack rỗng: IsEmpty (S)

Khi sử dụng các thao tác lấy phần tử trong stack ra thì việc trước tiên ta cần kiểm tra xem stack đó có rỗng hay không? Nếu rỗng thì không thể lấy các phần tử ngược lại thì được phép lấy các phần tử.

```
bool IsEmpty(STACK s){
	if (s.top == NULL){
		return true;
	}
	return false;
}
```

- Đưa phần tử vào Stack: Push (S,x)

```
void Push (STACK &s, int x){
	//tao mot node moi va truyen vao du lieu x
	NODE *NewNode = CreateNode(x);
	//neu node moi khoi tao thanh cong
	if (NewNode != NULL){
		//neu stack ban dau dang rong
		if (IsEmpty(s)){
```

```
                    // s.top = NewNode
                    s.top = NewNode;
                }
                else{ //nguoc lai them NewNode vao dinh cua stack
                NewNode->next = s.top;
                s.top = NewNode;
                }
        }
}
```

- ## Lấy phần tử ra khỏi Stack: Pop (S, x)

  Thao tác pop với stack khi cài đặt bằng danh sách liên kết chính là việc lấy các node từ trong top của stack đó ra và đồng thời thực hiện xóa node đó khỏi stack.

```
int Pop (STACK &s){
        //tao node p
        NODE *p;
        //tao bien x de gan bang du lieu cua node
        int x;
        //neu stack khong rong
        if (!IsEmpty(s)){
                //gan node p bang phan tu top cua stack
                p = s.top;
                //thay doi lai top cua stack
                s.top = p->next;
                //gan du lieu cua node p vao x
                x = p->data;
                //xoa di node p vua lay
                delete p;
                //tra ve x
                return x;
```

```
        }
    }
```

- Nhập xuất node vào stack

Hàm nhập:

```
void Input(STACK &s, int n){
        //duyen tu 0 den N
        for(int i = 0; i < n; i++){
        //tao bien x va nhan gia tri tu ban phim
        int x;
        cout << "NHAP SO THU  " << i << endl;
        cin >> x;
        cout << endl;
        //truyen stack va x vua nhap vao ham Push de dua vao stack
        Push(s,x);
        }
}
```

Hàm xuất:

```
void Output(STACK s)
{
        //tao node p
        NODE *p;
        //duyet tu phan tu tren dinh top den phan tu cuoi stack
        for(p = s.top; p != NULL; p=p->next){
                //hien thi du lieu
                cout << p->data;
        }
}
```

Chương trình cài đặt stack bằng danh sách liên kết hoàn chỉnh

```
#include<iostream>
using namespace std;
struct node
{
	int data;
	node *next;
};
typedef struct node NODE;
struct stack
{
	NODE *top;
};
typedef struct stack STACK;
void Init (STACK &s){
	//gan top cua stack ve NULL
	s.top = NULL;
}
NODE* CreateNode (int x) {
	NODE *p;
	p = new NODE;
	//neu p = NULL thi ko du bo nho cap phat
	if (p==NULL) {
		cout << "KHONG DU BO NHO!";
		return NULL;
	}
	//gan data bang X
	p->data=x;
	//gan con tro next bang NULL
	p->next=NULL;
	//tra ve node p
	return p;
}
bool IsEmpty(STACK s){
```

```
	if (s.top == NULL){
		return true;
	}
	return false;
}
void Push (STACK &s, int x){
	//tao mot node moi va truyen vao du lieu x
	NODE *NewNode = CreateNode(x);
	//neu node moi khoi tao thanh cong
	if (NewNode != NULL){
		//neu stack ban dau dang rong
		if (IsEmpty(s)){
			// s.top = NewNode
			s.top = NewNode;
		}
		else{ //nguoc lai them NewNode vao dinh cua stack
			NewNode->next = s.top;
			s.top = NewNode;
		}
	}
}
int Pop (STACK &s){
	//tao node p
	NODE *p;
	//tao bien x de gan bang du lieu cua node
	int x;
	//neu stack khong rong
	if (!IsEmpty(s)){
		//gan node p bang phan tu top cua stack
		p = s.top;
		//thay doi lai top cua stack
		s.top = p->next;
		//gan du lieu cua node p vao x
```

```
                x = p->data;
                //xoa di node p vua lay
                delete p;
                //tra ve x
                return x;
        }
}
void Input(STACK &s, int n){
        //duyen tu 0 den N
        for(int i = 0; i < n; i++){
        //tao bien x va nhan gia tri tu ban phim
        int x;
        cout << "NHAP SO THU  " << i << endl;
        cin >> x;
        cout << endl;
        //truyen stack va x vua nhap vao ham Push de dua vao stack
        Push(s,x);
        }
}
void Output(STACK s)
{
        //tao node p
        NODE *p;
        //duyet tu phan tu tren dinh top den phan tu cuoi stack
        for(p = s.top; p != NULL; p=p->next){
                //hien thi du lieu
                cout << p->data;
        }
}
int main(){
        STACK s;
        //khoi tao stack
        Init(s);
```

```
    //nhap so n
    int n;
    cout << "NHAP N: ";
    cin >> n;
    //nhap n phan tu vao stack
    Input(s,n);
    //hien thi cac phan tu vua nhap vao stack
    cout << "DANH SACH CAC PHAN TU VUA NHAP VAO STACK\n";
    Output(s);
    //push phan tu vao stack
    int x = 66;
    Push(s,x);
    //hien thi lai stack sau khi push
    cout << "STACK SAU KHI PUSH PHAN TU: \n" << x << endl;
    Output(s);
    //thuc hien lay phan tu top trong stack ra va xoa di
    int p = Pop(s);
    cout << "PHAN TU VUA DUOC LAY RA VA XOA DI LA " << p << endl;
    //stack sau khi pop
    cout << "\nSTACK SAU KHI POP LA\n";
    Output(s);
}
```

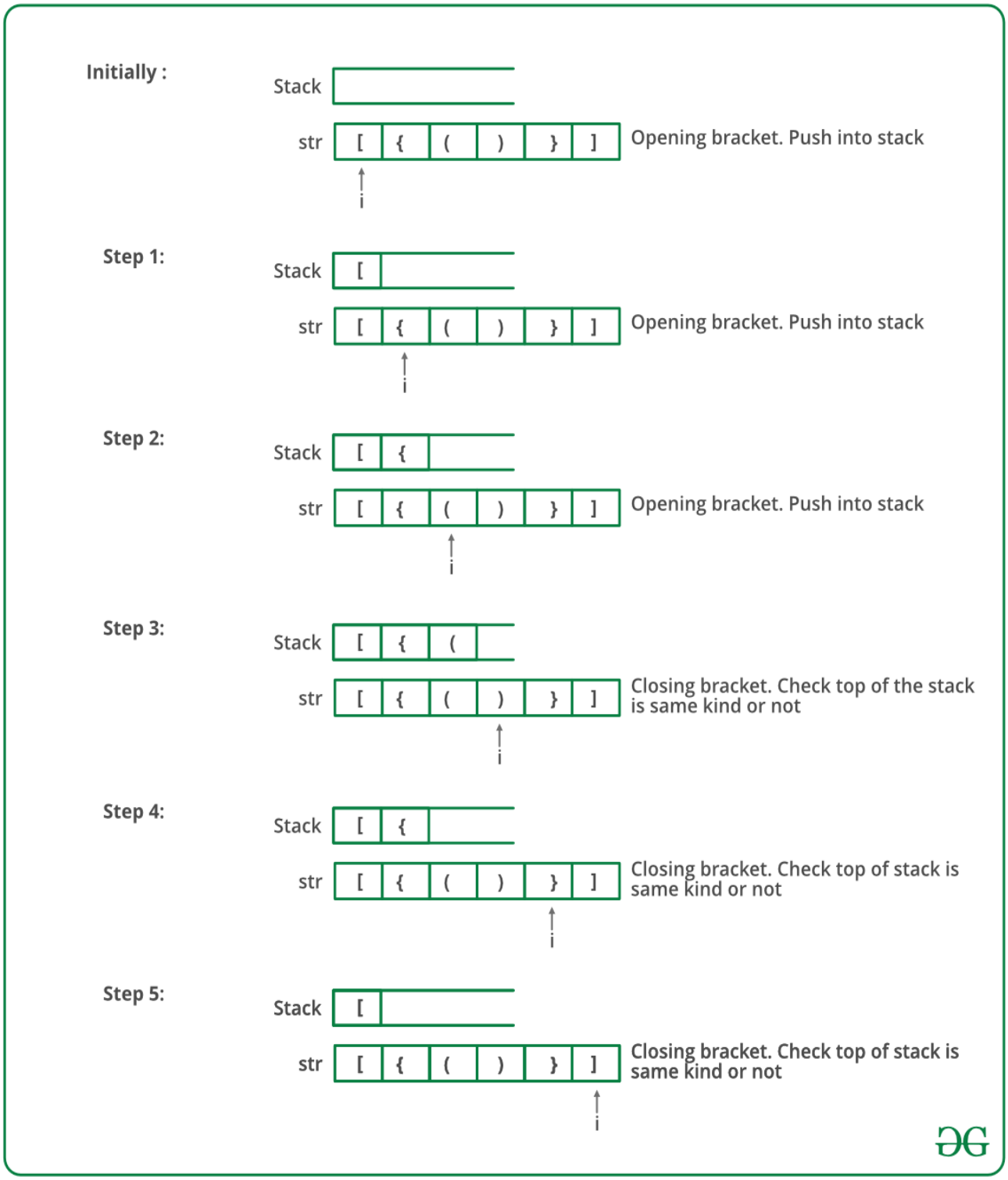
BÀI TẬP ÁP DỤNG:

1. Cho xâu S gồm các ký tự "(", "{", "[", "]", "}", ")"
   Hãy xây dựng và sử dụng cấu trúc ngăn xếp (stack) với các yêu cầu sau:
   a. Định nghĩa cấu trúc ngăn xếp để lưu trữ xâu S. (Có cấu trúc mẫu đã định nghĩa ở trên)
   b. Viết các hàm thao tác với cấu trúc ngăn xếp trong câu 1a: push, pop, kiểm tra stack rỗng, lấy phần tử ở đỉnh của stack, đếm số phần tử trong stack. (Hàm thao tác được viết ở trên)

c. Viết hàm nhập vào xâu S gồm có N kí tự. Sử dụng cấu trúc và các hàm đã được định nghĩa ở câu 1a và 1b để kiểm tra xem S có phải dãy ngoặc đúng hay không? Kết quả trả về true nếu đúng và false nếu sai (Biết rằng cặp ngoặc được coi là phù hợp nếu dấu ngoặc mở xuất hiện ngay bên trái dấu ngoặc đóng).
Ví dụ mẫu:
(1) ({[)]}: sai
(2) (]: sai
(3))()[]: sai
(4) ()[]{}: đúng
(5) (){{[(([{(())}]))]}}: đúng

Ý tưởng:

Duyệt qua từng dấu ngoặc trong dãy ngoặc; Sử dụng một stack để *push* các dấu ngoặc mở vào stack, mỗi khi gặp dấu ngoặc đóng, thực hiện *pop* một phần tử khỏi stack. Dãy ngoặc sẽ không hợp lệ khi bạn không thể *pop* hoặc khi kết thúc duyệt mà stack vẫn chưa rỗng.


Initially :
Stack
str
[{ ( ) }]
Opening bracket. Push into stack
i
Step 1:
Stack
[
str
[{ ( ) }]
Opening bracket. Push into stack
i
Step 2:
Stack
[{
str
[{ ( ) }]
Opening bracket. Push into stack
i
Step 3:
Stack
[{ (
str
[{ ( ) }]
Closing bracket. Check top of the stack is same kind or not
i
Step 4:
Stack
[{
str
[{ ( ) }]
Closing bracket. Check top of stack is same kind or not
i
Step 5:
Stack
[
str
[{ ( ) }]
Closing bracket. Check top of stack is same kind or not
i

Hàm:

***Lưu ý: các hàm khởi tạo và hàm thao tác trong cấu trúc stack đã được viết ở trên thay kiểu dữ liệu thành char thay vì mẫu là int

```
//1c. Kiểm tra xem S có phải dãy ngoặc đúng hay không?
bool kiemTraChuoiNgoacCanBang(string chuoi)
{
    STACK s;
    // khoi tao stack
    Init(s);

        char x;

        // Traversing the Expression
        for (int i = 0; i < chuoi.length(); i++)
        {
                if (chuoi[i] == '(' || chuoi[i] == '['
                        || chuoi[i] == '{')
                {
                        // Push the element in the stack
                        Push(s,chuoi[i]);
                        continue;
                }

                // IF current current character is not opening
                // bracket, then it must be closing. So stack
                // cannot be empty at this point.
```

```
if (IsEmpty(s))
        return false;

switch (chuoi[i]) {
case ')':

        // Store the top element in a
        x = Pop(s);
        if (x == '{' || x == '[')
                return false;
        break;

case '}':

        // Store the top element in b
        x = Pop(s);
        if (x == '(' || x == '[')
                return false;
        break;

case ']':

        // Store the top element in c
        x = Pop(s);
        if (x == '(' || x == '{')
```

```
                return false;
            break;
        }
    }

    // Check Empty Stack
    return (IsEmpty(s));
}
```

2. Cho chuỗi S (ví dụ S=”ABCD”), hãy in ra màn hình chuỗi đảo ngược của chuỗi S (”DCBA”) bằng cách sử dụng cấu trúc ngăn xếp (stack). Hãy:
   a. Định nghĩa cấu trúc dữ liệu biểu diễn ngăn xếp theo yêu cầu. (Có cấu trúc mẫu đã định nghĩa ở trên)
   b. Viết hàm in ra chuỗi đảo ngược của chuỗi S và các hàm khác có liên quan, sử dụng cấu trúc dữ liệu trong câu 2.a.

Ý tưởng:

1) Tạo một ngăn xếp(stack) trống.

2) Từng bước một đẩy tất cả các ký tự của chuỗi vào ngăn xếp.

3) Từng bước một pop tất cả các ký tự từ ngăn xếp và đặt chúng trở lại chuỗi.

Hàm:

***Lưu ý: các hàm khởi tạo và hàm thao tác trong cấu trúc stack đã được viết ở trên thay kiểu dữ liệu thành char thay vì mẫu là int

```
//2b. Dao nguoc chuoi
```

```
string daoNguocChuoi(string chuoi){
    STACK s;
    // khoi tao stack
    Init(s);
    string kq = "";
        for (int i = 0; i < chuoi.length(); i++)
        {
            Push(s,chuoi[i]);
        }

        while(!IsEmpty(s)){
            kq = kq + Pop(s); //them 1 ky tu vao chuoi bang toan tu +
        }
        return kq;
}
```

3. Viết chương trình nhập vào một số nguyên không âm bất kỳ, sau đó hiển thị lên màn hình số đảo ngược thứ tự của số nguyên vừa nhập vào (ví dụ: nếu nhập số 12345, hiển thị lên màn hình số 54321) bằng cách sử dụng ngăn xếp stack.

Ý tưởng:

Cho số n = 12345

Đầu vào: 12345
Đầu ra: 54321

Bắt đầu duyệt và lưu trữ lần lượt các chữ số của số đã cho trong ngăn xếp và cập nhật số dưới dạng số / 10.

- bước 1: n = 1234 ngăn xếp = 5
- bước 2: n = 123 ngăn xếp = 5,4
- bước 3: n = 12 ngăn xếp = 5,4,3
- bước 4: n = 1 ngăn xếp = 5,4,3,2
- bước 5: n = stack = 5,4,3,2,1

Sau đó, bật các phần tử từ ngăn xếp và tạo thành số dưới dạng:

reverse = reverse + (giá trị ở đầu ngăn xếp * i) (trong đó i = 1, cập nhật i là i * 10 ở mỗi bước)

Do đó, kết quả = 54321

Hàm:

***Lưu ý: các hàm khởi tạo và hàm thao tác trong cấu trúc stack đã được viết ở trên

```
//3b. Dao nguoc so
int daoNguocSo(int so){
    STACK s;
    // khoi tao stack
    Init(s);

    int kq = 0;
    int i = 1;
    while(so!=0)
        {
            int tam = so%10;
            Push(s,tam);
            so = so/10;
```

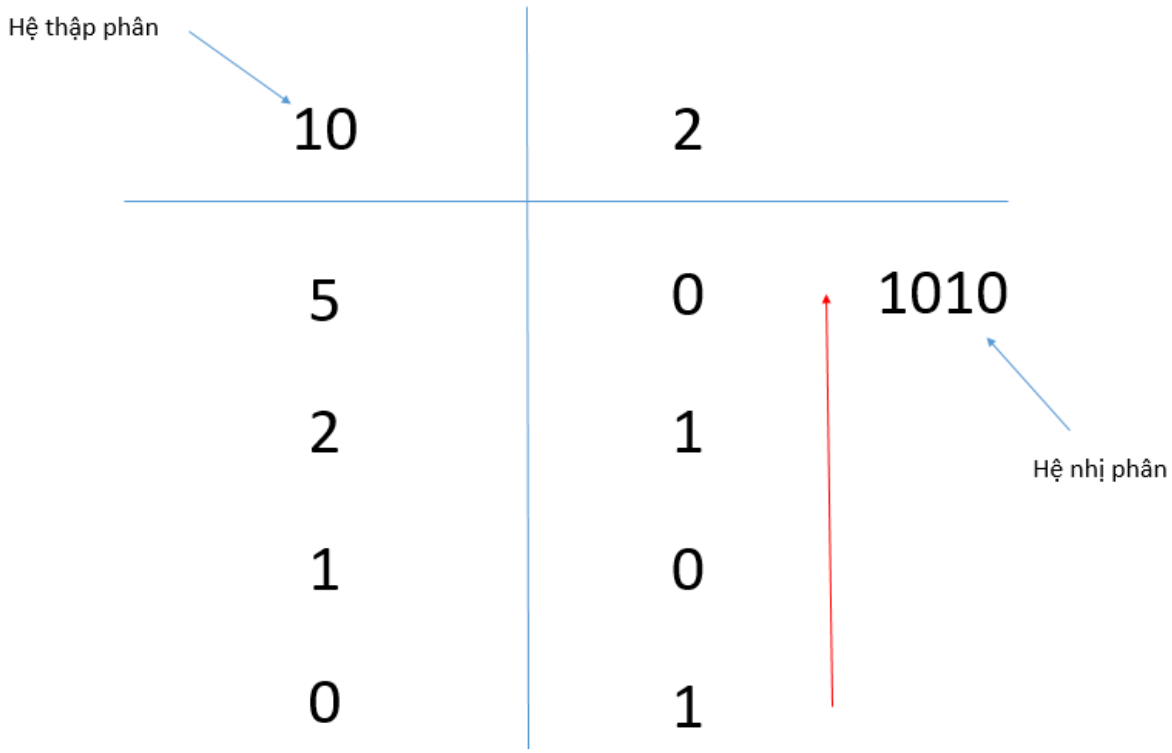
```
        }

    while(!IsEmpty(s)){
        int num = Pop(s);
        kq= kq + num*i;

        i=i*10;
    }

        return kq;
}
```

4. Viết chương trình chuyển đổi một số nguyên N trong hệ thập phân (hệ 10) sang biểu diễn ở các hệ sau
    a. Hệ nhị phân (hệ 2), sử dụng ngăn xếp stack.
    b. Hệ thập lục phân, Sử dụng ngăn xếp stack

Ý tưởng:

Ví dụ: ta có một số nguyên 10 là số thập phân, khi chuyển sang số nhị phân sẽ là 1010.

Vậy làm sao nó có thể chuyển được như vậy, đơn giản chỉ là lấy số thập phân đó và chia lấy dư cho 2, khi đó các con số phần dư được lấy ngược từ dưới lên chính là hệ nhị phân.



Như các bạn đã biết thì Stack hoạt động theo quy tắc LIFO (last in first out), lưu nó vào Stack rồi lấy nó ra theo quy tắc LIFO là xong. Cứ mỗi lần chia lấy dư như vậy ta sẽ lưu vào Stack, cho đến khi số chia bằng 0 thì ta thực hiện lấy phần tử đầu (top) trong Stack ra, như vậy dãy số được lấy ra chính là dãy nhị phân.

Hàm:

***Lưu ý: các hàm khởi tạo và hàm thao tác trong cấu trúc stack đã được viết ở trên

```
//4. Thap phan sang nhi phan
string thapPhanSangNhiPhan(int so){
    STACK s;
   // khoi tao stack
   Init(s);
```

```
    string kq = "";

    while(so!=0)
          {
             int tam =  so%2;
             Push(s,tam);
             so = so/2;

          }

    while(!IsEmpty(s)){
       int num = Pop(s);
       kq= kq + num;
    }

          return kq;
}
```

5. Hãy viết chương trình mô phỏng cho bài toán “Tháp Hà Nội” bằng cách sử dụng ngăn xếp. (bài tập cộng điểm)