1. Cài đặt Stack :

Cách 1 : Dựa vào mảng :

#include<bits/stdc++.h>

using namespace std;

const int max\_size = 100;

int stackk[100];

int top=0;

void push(int value) {

    if (top<max\_size) {

        stackk[top]=value;

        top++;

    }

    else cout << "stack overflow !" << endl;

}

int pop() {

    if (top>0) {

        return stackk[--top];

    }

    else {

        cout << "Stack overflow !" << ' ';

        return -1;

    }

}

bool isEmpty() {

    return top==0;

}

bool isFull() {

    return top==max\_size;

}

int main () {

    push(1);

    push(2);

    push(3);

    cout << pop() << '\n'; // lấy phần tử trên cùng của stack (3)

    cout << pop() << '\n'; // lấy phần tử trên cùng của stack (2)

    cout << pop() << '\n'; // lấy phần tử trên cùng của stack (1)

    cout << pop() << '\n'; // stack underflow, trả về giá trị mặc định (-1)

    system("pause");

    return 0;

}

Cách 2 : Dựa vào danh sách liên kêt :

#include <iostream>

class Node {

public:

    int value;

    Node\* next;

    Node(int value) {

        this->value = value;

        next = nullptr;

    }

};

class Stack {

public:

    Node\* top;

    Stack() {

        top = nullptr;

    }

    ~Stack() {

        while (!isEmpty()) {

            pop();

        }

    }

    void push(int value) {

        Node\* node = new Node(value);

        if (isEmpty()) {

            top = node;

        } else {

            node->next = top;

            top = node;

        }

    }

    int pop() {

        if (!isEmpty()) {

            int value = top->value;

            Node\* temp = top;

            top = top->next;

            delete temp;

            return value;

        } else {

            std::cout << "Stack underflow" << ' ';

            return -1;

        }

    }

    bool isEmpty() {

        return top == nullptr;

    }

};

int main() {

    Stack stack;

    stack.push(1);

    stack.push(2);

    stack.push(3);

    std::cout << stack.pop() << '\n'; // lấy phần tử trên cùng của stack (3)

    std::cout << stack.pop() << '\n'; // lấy phần tử trên cùng của stack (2)

    std::cout << stack.pop() << '\n'; // lấy phần tử trên cùng của stack (1)

    std::cout << stack.pop() << '\n'; // stack underflow, trả về giá trị mặc định (-1)

    system("pause");

    return 0;

}

1. Bài tập ứng dụng Stack :

DSA07001 : Ngăn Xếp 1 :

Table

Description automatically generated with medium confidence

Phân tích tính phù hợp :

Bài toán yêu cầu sử dụng ngăn xếp để lưu trữ các giá trị nguyên và thực hiện các thao tác push, pop và show trên stack. Với điều kiện đề bài, tổng số phần tử trong stack không vượt quá 200, do đó sử dụng stack là phù hợp. Trong quá trình giải bài toán, chúng ta sẽ sử dụng stack để lưu trữ các giá trị nguyên, và thực hiện các thao tác push, pop và show trên stack. Khi gặp lệnh push, chúng ta sẽ đưa giá trị vào đỉnh của stack. Khi gặp lệnh pop, chúng ta sẽ xóa giá trị đầu tiên trong stack. Khi gặp lệnh show, chúng ta sẽ in ra tất cả các giá trị đang có trong stack theo thứ tự lưu trữ. Do đó, sử dụng stack là phù hợp với bài toán này.

Ứng dụng :

Trong bài toán trên, stack được sử dụng để lưu trữ các phần tử được thêm vào (thông qua thao tác push) và xóa phần tử ở đỉnh (thông qua thao tác pop). Khi gặp lệnh show, ta sử dụng stack để in ra các phần tử hiện có trong stack.

Code :

#include<bits/stdc++.h>

using namespace std;

int main () {

    string s;

    stack<int> st;

    while (cin >> s) {

        if(s=="push"){

            int a;

            cin >> a;

            st.push(a);

        }

        else if (s=="pop") {

            if (!st.empty()) st.pop();

        }

        else {

            stack<int> st2=st;

            vector<int> v;

            if (st2.empty()) {

                cout << "empty" << endl;

                break;

            }

            while (!st2.empty()) {

                v.push\_back(st2.top());

                st2.pop();

            }

            for (int i=v.size()-1;i>=0;i--) cout << v[i] << " ";

            cout << endl;

        }

    }

    return 0;

}