**­BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI: QUẢN LÝ ĐẶT PHÒNG**

**Giáo viên hướng dẫn: Th.sy Đậu Hải Phong**

**Sinh viên thực hiện:**  Nguyễn Trung Thành

**Hà Nội, 2022**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI: QUẢN LÝ ĐẶT PHÒNG**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1671020295 | Nguyễn Trung Thành | 12/10/2004 |  |  |

### 

### CÁN BỘ CHẤM THI

**Hà Nội, 2022**

**LỜI NÓI ĐẦU**

Sau quá trình học tập, rèn luyện kỹ năng, tư duy và cách xử lý các trường hợp trong môn Cấu trúc dữ liệu và giải thuật em được thầy Đậu Hải Phong hướng dẫn rất tận tình . Thầy đã chia sẻ những kiến thức phức tạp trở nên đơn giản dễ hiểu , thầy đã truyền cho bọn em động lực để lập trình , làm đánh thức ngọn lửa tư duy lập trình. Em đã ngồi phác thảo ý tưởng và cho ra một chương trình quản lý đặt phòng khách sạn bằng ngôn ngữ C++ . Đề tài của là Quản Lý Đặt Phòng khách sạn .

Đến với bài báo cáo này , em làm về quản lý đặt phòng khách sạn là sẽ chia các phòng có các thuộc tính của đối tượng phòng . Các thuộc tính này gồm có mã phòng , tên phòng , giá phòng , người mượn phòng , ngày mượn phòng , kết thúc trả phòng . Các thuộc tính này tồn tại để những người lập trình viết những đoạn mã code cho máy tính hiểu được ,và giúp chúng có thể biên dịch từ ngôn ngữ lập trình sang một file exe hoặc một file text nào đó . Ở bài báo cáo này em sẽ biên dịch sang file text có tên là QuanLyDatPhong.txt . Và từ file này em có thể truy xuất hiển thị các thông tin , ghi các thông tin vào file , và có các chức năng xây dựng cho đề tài .

**MỤC LỤC**

**(Đánh tự động với 3 mức)**

**MỤC LỤC HÌNH ẢNH**

**(Đánh tự động nếu có)**

**MỤC LỤC BẢNG**

**(Đánh tự động nếu có)**

# Chương 1 Giới thiệu đề tài và cơ sở lý thuyết

## 1.1 Giới thiệu đề tài phát biểu bài toán

Đây là đề tài quản lý đặt phòng khách sạn , được xây dựng dựa trên cấu trúc stack cách thức hoạt động chính là ( LIFO) tức là last in first out . Ý tưởng của đề tài là xây dựng đoạn chương trình quản lý số tầng và phòng trong khách sạn .

Stack hay còn được gọi là ngăn xếp , chúng ta có thể tưởng tượng ra những chiếc đẽ xếp chồng lên nhau , đúng vậy mỗi một chiếc đĩa chúng ta hình dung ra nó là các node . Ngăn xếp được ứng dụng phổ biến trong nghành công nghệ , ta ví dụ sử dụng nút ctrl + z và ctrl + y , chính xác 2 nút này dựa trên tư tưởng chính push và pop các stack với nhau .

Từ ý tưởng trên em đã xây dựng đề tài quản lý danh sách phòng trong khách sạn , danh sách phòng này gồm các thuộc tính phổ biến như sau : id ( mã phòng ), tên phòng , giá thuê , người thuê , ngày thuê , hạn nộp tiền phòng .

Để ngăn xếp có các tính năng ưu việt cho người dùng dễ dàng quản lý cho khách sạn của mình , em đã xây dựng những chức năng đa dạng giúp cho người dùng có thể quản lý một các dễ dàng . Những đoạn mã em xây dựng bằng ngôn ngữ C++ và không có giao diện . Về căn bản đây là những đoạn mã code thuần , từ đầu đến cuối không hàm sẵn có , và được lên ý tưởng nhưng chức năng của hàm một cách ngẫu nhiên khi viết đoạn mã . Ngẫu nhiên nhưng có đánh giá và sử dụng thử thấy hợp lý và triển khai tốt dành cho người dùng.

## 1.2 Cơ sở lý thuyết con trỏ , cấu trúc , stack

### 1.2.1 Con trỏ

Giới thiệuc cơ bản qua

### 1.2.2 Cấu trúc

Giới thiệu cơ bản qua

### 1.2.3 Cấu trúc Stack

Là gì

ứng dụng của nó

# Chương 2 Phân tích đề tài

## 2.1 Dữ liệu lưu trữ

## 2.2 Cấu trúc dữ liệu

Cấu trúc Node : ý nghĩa đây là các node trong ngăn xếp , ta nhìn thấy hình dưới có thể nói các ô hình chữ nhật đại diện cho mỗi node .

Bây giờ ta xem cấu trúc khai báo node :

*struct Node{*

*PhongKhachSan dataKs;*

*Node \*next;*

*};*

Giải thích : ta sử dụng kiểu dữ liệu là struct ( hay còn gọi cấu trúc) bên trong node có thông tin dữ liệu của đối tượng là dataKs ( ý nghĩa data khách sạn )

Cùng xem data khách sạn có những gì :

*struct PhongKhachSan{*

*int id;*

*string name;*

*float giaThue;*

*};*

*Gồm thuộc tính id , tên , giaThue ,... sẽ còn thêm .*

Tiếp đến là cấu trúc stack ( ngăn xếp )

struct stack{

Node \*top;

};

Ý nghĩa : Cấu trúc stack này sẽ có thông tin của node , với việc sử dụng cấu trúc node ta có thể truy cập thông tin node qua node top ( ý nghĩa là node đầu tiên của stack hay còn gọi là đỉnh )

## 2.3 Chức năng hàm

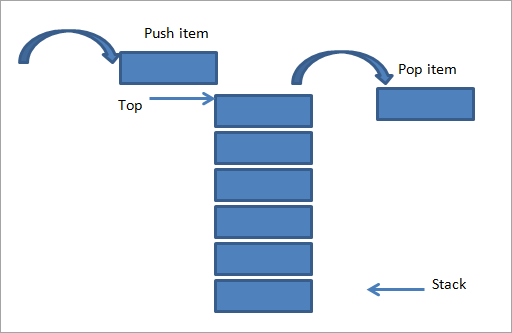
Dưới đây là một số ý tưởng xây dựng hàm có tính ứng dụng cao hơn

### 2.3.1 Hàm push

Ý nghĩa thêm vào , từ phần tử đối tượng , ta sẽ thêm phần tử đối tượng vào bên trong stack

Tại sao cần nó , push là một trong những chức năng quan trọng trong stack ,nó giúp cho các node được thêm vào bên trong stack ( ngăn xếp ) , ví dụ ở bên dưới ta có thể thấy là một một ngăn xếp chứa 5 node ( chưa kể một node đang được push vào ) . Đúng vậy lúc này ngăn xếp có 5 node , em sẽ thêm một node vào bằng hàm push , những thông tin được di chuyển và nối lại các node với nhau .Ta có nối các node với nhau không , nhưng về bản chất stack khác linked list ở điểm là stack 2 chức năng chính thêm vào và bớt đi xong xóa .

Nó sử dụng như nào ? Push là các chúng ta thêm vào , vậy sử dụng ra sao . Ta hình dung rằng mỗi node cần nối với những node tiếp theo , nó chỉ dừng lại khi node tiếp theo trỏ đến NULL , tức là ô nhớ của node không được lưu địa chỉ gì trong ô nhớ đệm của máy tính .



### 2.3.2 Hàm Pop

ý nghĩa là xóa đi phần tử đối tượng ( tại sao gọi là phần tử đối tượng , ta thấy node này nó không chỉ chưa phần tử số nguyên , mà nó còn có cả float , string , date time ... nên gọi chung là đối tượng đi thì sẽ có ý nghĩa nhiều thuộc tính khác nhau trong đối tượng ) .

Pop sẽ xóa node đầu tiên trong ngăn xếp và sẽ được xóa đi trong stack , quay lại bên trên cấu trúc stack ta thấy được stack .

Pop được sử dụng như nào ? Pop sử dụng rất phổ biến liên quan đến stack , có thể nói mọi hàm gần như đều xây dựng trên tư tưởng là stack . Chỉ đơn giản là nó sẽ đưa node ở đỉnh ra bên ngoài và làm cắt đắt điểm nối giữa chúng với các node còn lại trong stack , và node được đưa ra ngoài sẽ trả về phần tử đối tượng .

Chưa kể một số hàm cơ bản , mà cấu trúc strack bắt buộc có các thuộc tính cấu trúc node và trong node lại có thuộc tính cấu trúc đối tượng . Vậy nên khi ta sẽ đi nó đồng thời sẽ mất tất dữ liệu bên trong ô nhớ của node đấy .

### 2.3.3 Hàm chia thành các tầng , mỗi tầng sẽ là 1 stack .

Ý tưởng này dựa trên việc xây dựng quản lý một khách sạn , sẽ gồm có rất nhiều tầng , mỗi tầng lại có mức giá , và mức sống khác nhau .

Từ ý tưởng thực tế em xây dựng mã chương trình như sau :

Mã chương trình stack lại dựa trên ý tưởng sử dụng mảng , ý nghĩa luồng hoạt động mảng sẽ chia mặc định các vị trí cố định trong ô nhớ chỉ số , lúc này em sẽ thêm từng stack ( đại diện cho mỗi từng vào bên trong chỉ số của mảng)

Ví dụ ban đầu đưa , em có tệp dữ liệu là số nguyên sau đó chuyển các tệp số nguyên này vào trong mỗi ngăn xếp , mỗi ngăn xếp này sẽ được trải chiều cao như một ngọn tháp , một ngọn tháp cao quá thì dữ liệu dễ loãng thông tin và khó quản lý . Em sẽ chia dữ liệu ngọn tháp nay ra thành các phần , lúc này em sử dụng đến stack kết hợp mảng

|  |  |  |  |
| --- | --- | --- | --- |
| Tâng1 | Tâng2 | Tâng3 | Tâng4 |
| Id:1 phong123 9000 | Id:5 phong223 8000 | Id:9 phong323 7000 | Id:13 phong424 6000 |
| Id:2 phong124 9000 | Id:6 phong224 8000 | Id:10 phong324 7000 | Id:14 phong425 6000 |
| Id:3 phong125 9000 | Id:7 phong225 8000 | Id:11 phong325 7000 | Id:15 phong426 6000 |
| Id:4 phong126 9000 | Id:8 phong226 8000 | Id:12 phong326 7000 | Id:16 phong427 6000 |

### 2.3.4 Hàm so sánh các mã id với nhau

Ý tưởng nếu mã id nào đã có rồi thì không được nhập vào stack nữa và thông báo nhập lại

Phác thảo ý tưởng : pop từng stack , nếu id nào đã tồn tại thì sẽ cho người dùng nhập lại ( ta sử dụng do while ) nếu id đúng rồi thì push vào 1 stack ( stack này sẽ là 1 tầng trong khách sạn

1. Góp ý về output() nhớ pop thì sau đó push
2. Đang đúng theo ý tưởng tôi mong muốn là nó sẽ hiển thị cả 2 stack

### 2.3.5 Hàm xây dựng thuật toán để sắp xếp giá phòng tăng từ thấp đến cao



*Step by step thuật toán selection sort*

Ý tưởng thuật toán là : Duyệt từ vị trí bắt đầu đến vị trí có độ dài của mảng , sau mỗi lần duyệt thuật toán sẽ tìm ra giá trị min , sau đó giá trị min đó sẽ đổi chỗ với phần tử đầu tiên và phần tử tiếp theo của mảng và nó sẽ tăng vị trí duyệt lên , để đến khi nào vị trí k-1 ( k ở đây nghĩa là vị trí cuối ) . Nhìn vào ví dụ hình ảnh trên đưa ra từng bước giải của thuật toán.

Sau khi xem xét triển khai vào code tôi thấy , vẫn phải gộp tất cả các tầng 1 , tầng 2 , ... vào 1 stack chung là khachSan

Ý tưởng triển khai là , sau mỗi lần duyệt nó sẽ xóa đi thằng pop

# Chương 3 Phân tích chương trình và triển khai mã chương trình

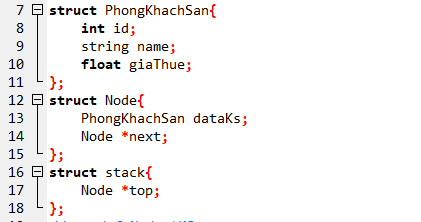
## 3.1 Thư viện

*#include<iostream>*

*#include<fstream>*

*#include <string>*

## 3.2 Cấu trúc



## 3.3 Khởi tạo

*void Init(stack &s) {*

*s.top = NULL;*

*}*

## 3.4 Kiểm tra

Ý nghĩa sử dụng kiểm tra stack có rỗng không , tức là con trỏ top có quản lý ô nhớ nào trong máy tính hay không .

Ứng dụng để kiểm tra stack có tồn tại hay không .

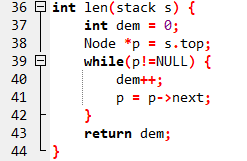
*int isEmpty(stack &s) {*

*return (s.top==NULL) ;*

*}*

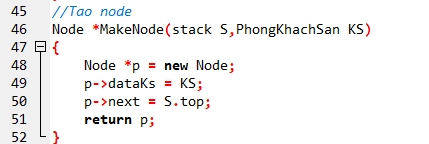
Ý nghĩa : hàm sẽ khởi tạo một stack rỗng .

## 3.5 Đếm độ dài stack

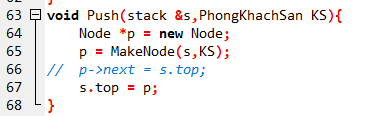


## 3.6 Tạo một node

Ý nghĩa sẽ tạo ra một node và nó sẽ được gọi lại trong các hàm như muốn thêm vào đầu , giữa , cuối.

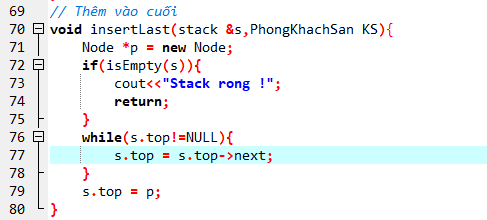


## 3.7 Thêm vào đầu



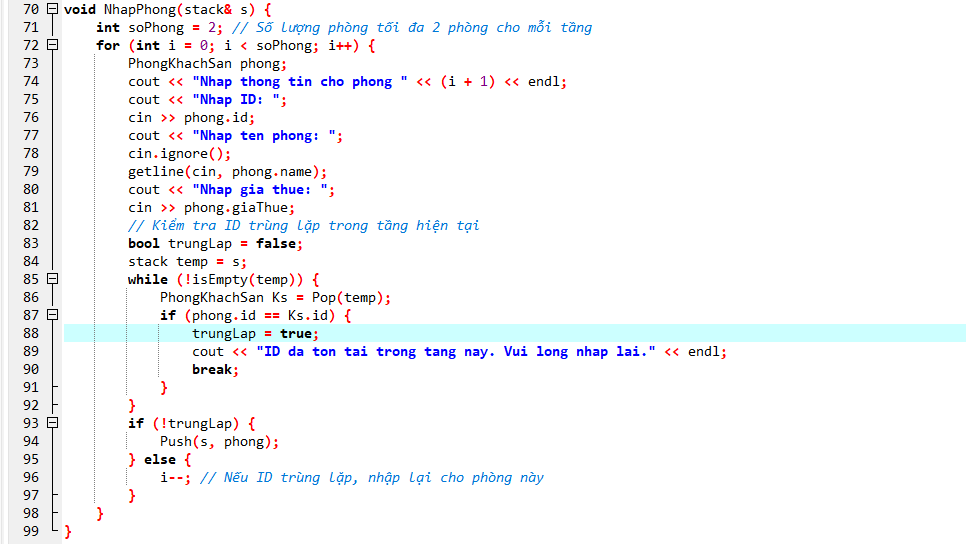
## 3.8 Thêm vào đỉnh

Ý tưởng là duyệt đến vị trí đầu , rồi gán vị trí đầu vào vị trí đầu .



## 3.9 Nhập dữ liệu phòng

Ý tưởng tôi sẽ sử dụng nhập số phòng thứ j , nếu số id phòng thứ j trùng với số id phòng thứ j kế tiếp thì nó sẽ bảo người dùng nhập lại .



3.4 Thêm vào đầu danh sách ( Push )

*//push vào*

*void push(stack &s,PhongKhachSan PhongKS)*

*{*

*Node \*p = new Node; // cap phat node p*

*p->dataKs = PhongKS;*

*p->next = s.top;*

*//tra ve*

*s.top = p; // gan vao top cua stack*

*}*

## 3.10 In ra màn hình sử dụng stack , và stack kết hợp mảng

### 3.5.1 Chưa kết hợp mảng

Cùng xem đoạn mã dưới đây :

void print(stack &s) {

stack temp = s;

while(!isEmpty(temp)) {

PhongKhachSan KS = Pop(temp);

cout<<KS.id<<"\t"<<KS.name<<"\t"<<KS.giaThue;

cout<<endl;

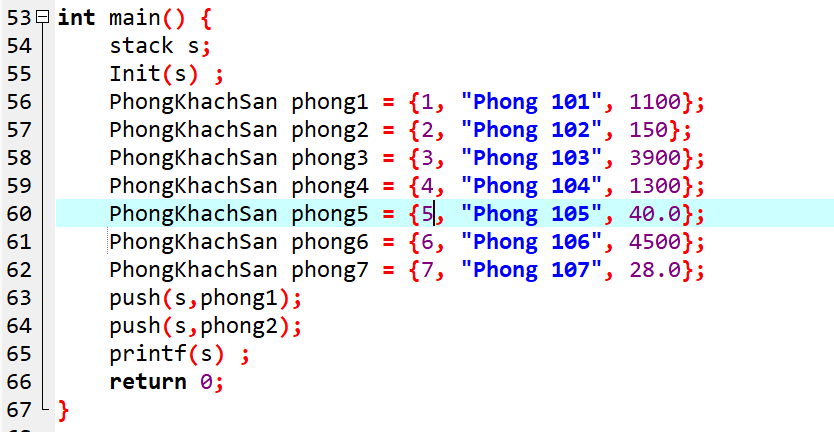
// Push(s,KS);

}

}

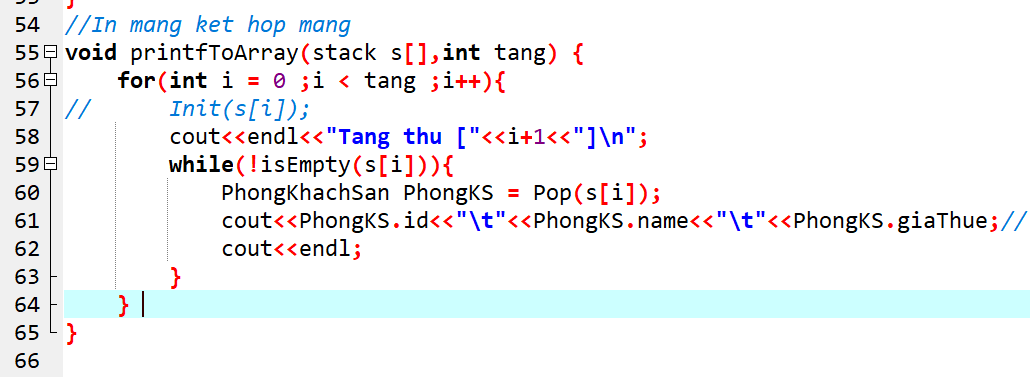
Bên trên là cách hiện phòng không sử dụng mảng , nó khá là đơn giản , chỉ dựa trên tư tưởng là lấy ra và xóa đi

Dưới đây là code ở hàm main gọi và in ra màn hình :



### 3.5.2 kết hợp mảng

### Ý tưởng khi tôi sử dụng hàm này nó sẽ in ra màn hình thông tin số tầng và trong mỗi số tầng sẽ có số phòng thứ j .



In ra màn hình sử dụng mảng , đây là kĩ thuật kết hợp mảng và stack

Về giải thuật nó tương tự như không kết hợp mảng:

Duyệt từng phần tử trong vị trí của mảng thứ i , mỗi vị trí tương ứng với mỗi stack , và mỗi stack này lại chữa những node trỏ đến ô bộ nhớ .

## 3.11 Lấy ra phần tử cuối ( Pop )

*// ham pop , lay ra phan tu cuoi*

*PhongKhachSan Pop(stack &s) {*

*stack temp = s;*

*PhongKhachSan KS = s.top->dataKs;*

*s.top = s.top->next;*

*return KS;*

*}*

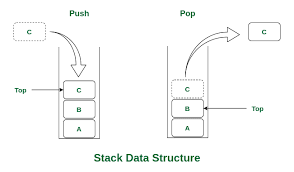
*Giải thuật code :*

Sử dụng một stack temp

Đối tượng phòng khách sạn được gán với đối tượng đầu của stack

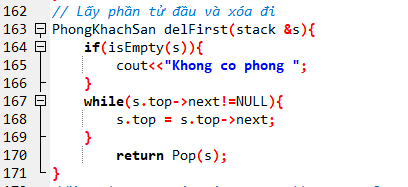
Sau đó gán đối tượng tiếp theo của stack là đỉnh của stack

Ý nghĩa : Xóa đi node đỉnh của stack hình ảnh minh họa :



1. Lấy ra phần tử đầu và xóa đi

Ý tưởng sử dụng lấy ra phần tử đầu tiên sau đó xóa đi để ứng dụng trong trường hợp muốn in ra theo thứ tự đảo ngược

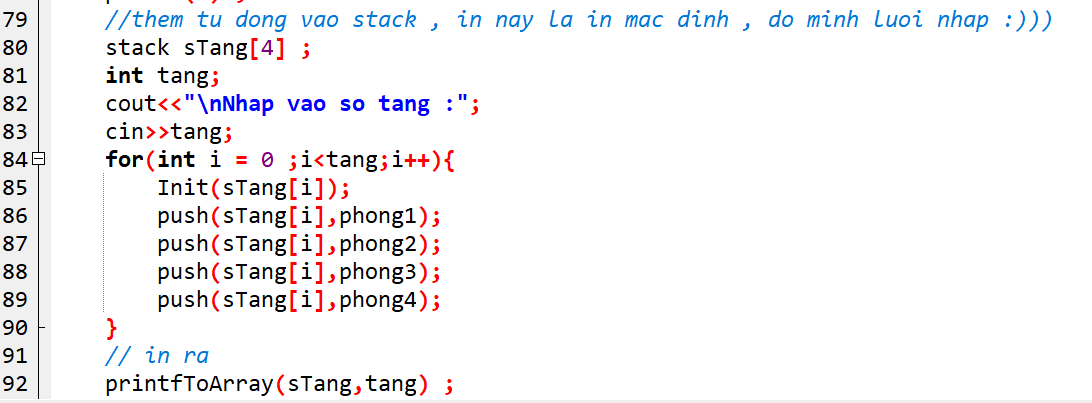


## 3.12 Điền thông tin nhập vào và điền mặc định

### 3.12.1 Điền mặc định

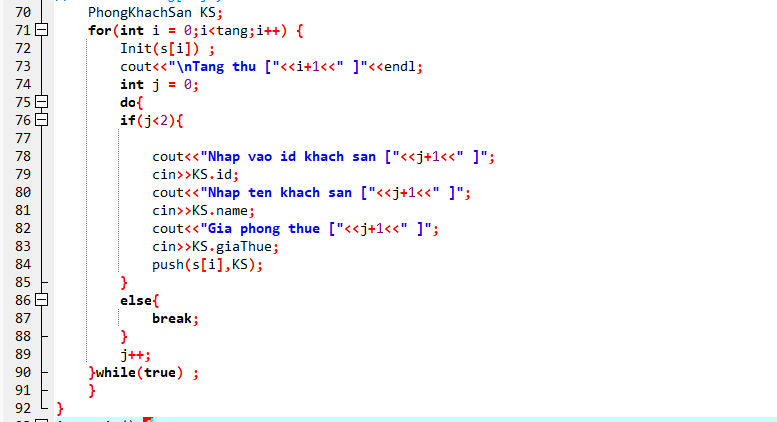
Điền thông tin mặc định thường sử dụng để cố định những giá trị thuộc tính , giúp lập trình viên tiết kiệm thời gian trong quá trình làm dự án , để lập trình viên không cần phải nhập từng dòng giá trị thuộc tính , điều đó khá là mất thời gian khi còn phải nghĩ ra cả tên nên nhập như nào

Khắc phục điều đó em sử dụng điền mặc định dưới hàm main :



### 3.12.2 Nhập thông tin

Người dùng nhập thông tin để quản lý phòng :



Giải thuật :

Duyệt qua từng vị trí , mỗi vị trí là một stack , mỗi stack là mỗi node trong ô nhớ .

Mỗi lần duyệt vị trí sẽ nhập vào từng node cho mỗi stack ( tầng 1 , 2 , 3 , 4 ) như đoạn code trên ý tưởng là mỗi tầng có 2 phòng . Sau đó sử dụng hàm push để thêm vào

## 3.13 Tìm kiếm trả về giá trị số nguyên để kiểm tra

Ý tưởng : sau khi tìm thấy id của phòng thì trả về giá trị 1 , nếu không tìm thấy trả về 0

Giải thuật :

Lấy ra từng phần tử trong stack , nếu phần tử lấy ra sau đó so sánh với id phòng cần tìm kiếm thì trả về 1 .

Nhap vao so tang :3

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

Nhap id phong tim kiem :2

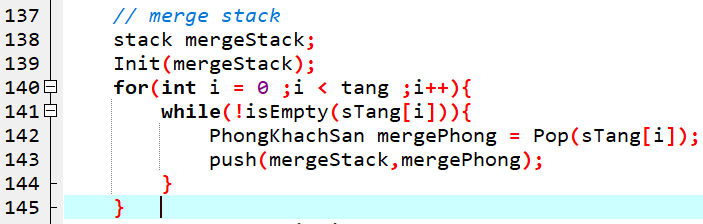
Tim thay phong co id la :2

## 3.14 Tìm kiếm trả về node phòng

Ý tưởng : Sau khi tìm kiếm id của phòng thì trả về node phòng đó .

## 3.15 Merge stack ( hợp nhất stack )

Ý tưởng là mỗi stack sẽ chứa trong mỗi vị trí trong mảng , bây giờ em sẽ merge từng node của mỗi stack vào bên trong stack có chung , ý nghĩa để hợp nhất tất cả cả node trong stack vào chung một stack , ứng dụng trong việc sắp xếp , tìm kiếm ,quy hoạch động .



Giải thuật :

Nó là sự kết hợp giữa pop và push

Lấy ra pop phần tử của stack thứ i thì push vào merge stack .

Kết quả mong muốn đã in ra thành công .

Hiển thị ra 5 tầng , mỗi tâng có đối số mặc định lần lượt id lần lượt từ 1-4

Nhap vao so tang :5

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

1 Phong 101 1100

2 Phong 102 150

3 Phong 103 3900

4 Phong 104 1300

Nhap id phong tim kiem :10000000

Khong tim thay !

Nhưng mà chúng ta thấy nó lại không sắp xếp theo đúng kiểu stack là từ trên xuống dưới , khái niệm stack là thêm vào cuỗi thì lấy ra đầu . Vậy thì in ra phải là 4 3 2 1

Lúc này ta có thể sử dụn đến vector

3.11 Sử dụng stack kết hợp vector

Ý tưởng pop các phần tử từ stack chung để push vào vector , nó sẽ giúp chúng ta in theo đúng thứ tự

Giải thuật :

Bây giờ phải xây dựng một hàm để push vào vector

Ý nghĩa để có thể in ra theo đúng thứ tự ngăn xếp . Nếu như ta chỉ sử dụng nguyên stack thì số vòng lặp là khá nhiều .

Lỗi ở bên trên là do em không push các phần tử vào vector , nó sẽ in ra theo kiểu ngược lại của stack tức là 1 2 3 4 , kết quả mong muốn phải là 4 3 2 1 .

Để xử lý trường hợp này ta xây dựng một hàm push các stack thứ i vào vector

Kết quả mong muốn đã in ra thành công .

Nhap vao so tang :5

4 Phong 104 1300

3 Phong 103 3900

2 Phong 102 150

1 Phong 101 1100

4 Phong 104 1300

3 Phong 103 3900

2 Phong 102 150

1 Phong 101 1100

4 Phong 104 1300

3 Phong 103 3900

2 Phong 102 150

1 Phong 101 1100

4 Phong 104 1300

3 Phong 103 3900

2 Phong 102 150

1 Phong 101 1100

4 Phong 104 1300

3 Phong 103 3900

2 Phong 102 150

1 Phong 101 1100

## 3.16 Hàm lấy phần tử đầu và xóa đi

Ý tưởng xây dựng hàm để lấy phần tử đàu ra bên ngoài , để khi thực hiện thao tác merge code có thể in theo thứ tự của stack là 4 3 2 1

Hàm xây dựng giống như hàm Pop nhưng ngược lại hàm này sẽ lấy ra phần tử đầu tiên và xóa đi trả về giá trị của x .

Bây giờ ta sẽ xây dựng thuật toán sắp xếp selection sort

Sau đó ta sẽ đánh giá mức độ phức tạp thuật toán O(n)

## 3.17Hàm tìm giá thuê phòng lớn nhất

Ý tưởng duyệt qua tất cả các node thứ i -1 rồi tìm vị trí nhỏ nhất . Tức là sau mỗi lần lần lặp ta sẽ cho node giảm đi ở vị trí sau node vừa duyệt rồi .

K này là vị trí thứ s.top->next nghĩa là node kế tiếp

## 3.18 Hàm tìm giá thuê phòng nhỏ nhất

## 3.14 Hàm sắp xếp selection sort

## 3.15 Hàm sắp xếp merge sort

## 3.16 Hàm tìm kiếm nhị phân

## 3.17 Hàm tìm kiếm chiều rộng

3.18 Hàm Ctrl z và Ctrl y xóa đi và phục hồi

3. Kết hợp stack với linked list

3 Kết hợp stack với queue