HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

Lập trình C cơ bản

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Lập trình C cơ bản

Tuần 8: Ứng dung ngăn xếp, hàng đợi

ONE LOVE. ONE FUTURE.

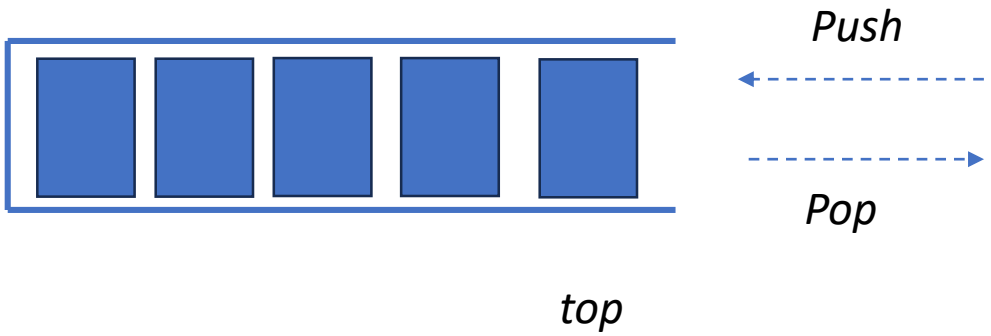
NỘI DUNG

- Bài tập 1: Mô phỏng ngăn xếp (P.03.08.01)
- Bài tập 2: Mô phỏng hàng đợi (P.03.08.02)
- Bài tập 3: Kiểm tra ngoặc (P.03.08.03)
- Bài tập 4: Những bình nước (P.03.08.04)

NGĂN XẾP VÀ HÀNG ĐỢI

Ngăn xếp:

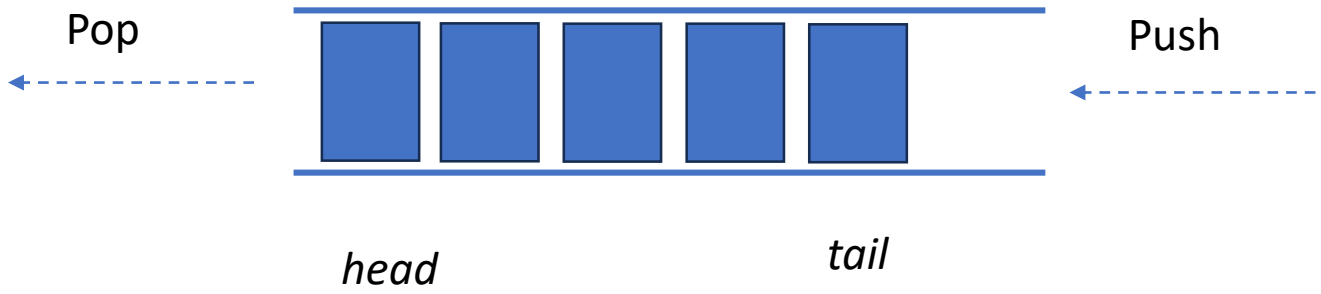
- Là một danh sách tuyến tính của các đối tượng
- Thao tác thêm mới và loại bỏ đều thực hiện ở một đầu (đỉnh hay top) của danh sách (Last-In-First Out)
- Một số thao tác cơ bản:
  - Push(x,S): Chèn 1 phần tử x vào ngăn xếp S
  - Pop(S): Lấy ra một phần tử khỏi ngăn xếp S
  - Top(S): Truy cập phần tử ở đỉnh của ngăn xếp
  - Empty(S): Trả về true nếu danh sách rỗng



NGĂN XẾP VÀ HÀNG ĐỢI

Hàng đợi:

- Là một danh sách tuyến tính của các đối tượng với 2 đầu head và tail
- Thao tác thêm mới được thực hiện ở một đầu (head) và loại bỏ được thực hiện ở đầu còn lại (tail) (First In First Out)
- Một số thao tác cơ bản:
  - Enqueue(x,Q) (Push): Chèn 1 phần tử x vào hang đợi S
  - Dequeue(S) (Pop): Lấy ra một phần tử khỏi hàng đợi S
  - Empty(S): Trả về true nếu hàng đợi rỗng
  - Top(S): tra cứu phần tử đầu của hàng đợi



BÀI TẬP 1: MÔ PHỎNG NGĂN XẾP (P.03.08.01)

- Thực hiện một chuỗi các thao tác trên 1 ngăn xếp, mỗi phần tử là một số nguyên:
  - Push v: Đẩy giá trị v vào ngăn xếp
  - Pop: Loại bỏ một phần tử ra khỏi ngăn xếp và in giá trị loại bỏ ra stdout (in ra NULL nếu ngăn xếp là rỗng)

- Dữ liệu:
  - Mỗi dòng là một thao tác thuộc một trong hai kiểu:
    - PUSH v
    - POP

- Kết quả:
  - Ghi kết quả của các thao tác POP, mỗi kết quả trên một dòng

BÀI TẬP 1: MÔ PHỎNG NGĂN XẾP

Ví dụ:

stdin	stdout
PUSH 1	3
PUSH 2	2
PUSH 3	5
POP	
POP	
PUSH 4	
PUSH 5	
POP	
#	

BÀI TẬP 1: MÔ PHỎNG NGĂN XẾP– MÃ GIẢ

- Dùng danh sách liên kết đơn (trỏ bởi *top*) để cài đặt ngăn xếp:
  - Hàm Pop tương ứng với loại bỏ phần tử đầu danh sách.
  - Hàm Push tương ứng với thêm một phần tử vào đầu danh sách.

```
struct Node{
    int value;
    struct Node* next;
}
```

```
makeNode(x){
   p = new Node();
   p -> value = x;
   return p;
}
```

```
Pop(){
if top==NULL return NULL;
x = top;
top = top-> next;
return x;
}
```

```
Push(x){
p = makeNode(x);
p->next = top;
top = p;
}
```

BÀI TẬP 1: MÔ PHỎNG NGĂN XẾP– CODE HOÀN CHỈNH

```
typedef struct Node{
    int value;
    struct Node* next;
}Node;

Node* top;// point to the top of the stack

Node* makeNode(int x){
    Node* p = (Node*)malloc(sizeof(Node));
    p->value = x; p->next = NULL;
    return p;
}
```

BÀI TẬP 1: MÔ PHỎNG NGĂN XẾP– CODE HOÀN CHỈNH

```
void initStack(){
    top = NULL;
}
int stackEmpty(){
    return top == NULL;
}
```

```
int  pop(){
    if(stackEmpty()) return ' ';
    int x = top->value;
    Node* p = top; top = top->next;
    free(p);
    return x;
}

void push(int x){
    Node* p = makeNode(x);
    p->next = top; top = p;
}
```

BÀI TẬP 1: MÔ PHỎNG NGĂN XẾP– CODE HOÀN CHỈNH

```
int main(){
    char cmd[0];
    while(1){
        scanf("%s",cmd);
        if(strcmp(cmd,"#")==0){
            break;
        }else if(strcmp(cmd,"PUSH")==0){
            int v; scanf("%d",&v);
            push(v);
        }else if(strcmp(cmd,"POP")==0){
            if(stackEmpty()){
                printf("NULL\n");
            }else{
                int v = pop();
                printf("%d\n",v);
            }}}
    return 0;}
```

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI (P03.08.02)

Thực hiện một chuỗi các thao tác trên một hàng đợi, mỗi phần tử là một số nguyên:

- PUSH v: Đẩy giá trị v vào hàng đợi
- POP: Loại bỏ một giá trị ra khỏi hàng đợi và in phần tử này ra stdout (in ra NULL nếu hàng đợi là rỗng)

- Dữ liệu:
  - Mỗi dòng là một thao tác thuộc một trong hai kiểu:
  - PUSH v
  - POP

- Kết quả:
  - Ghi kết quả của các thao tác POP (mỗi dòng cho một kết quả)

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI

Ví dụ 2.1:

stdin	stdout
PUSH 1	1
PUSH 2	2
PUSH 3	3
POP	
POP	
PUSH 4	
PUSH 5	
POP	
#	

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI

Ví dụ 2.2:

stdin	stdout
PUSH 1 POP POP PUSH 4 POP #	1 NULL 4

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI – MÃ GIẢ

- Dùng danh sách liên kết đơn với hai con trỏ *head* và *tail* để cài đặt hàng đợi:
  - Pop khỏi hàng đợi bằng cách lấy ra ở *head*
  - Push vào hàng đợi bằng cách đưa vào *tail*

```
struct Node{
    int value;
    struct Node* next;
}
```

```
makeNode(x){
   p = new Node();
   p -> value = x;
   return p;
}
```

```
Pop(){
if head = tail = NULL return ‘’;
v=head->value
head = head->next
return v
}
```

```
Push(x){
p = makeNode(x)
if head=tail=NULL then
head=tail=p; return;
tail->next = p
tail = p
return
}
```

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI – CODE HOÀN CHỈNH

```
typedef struct TNode{
    int value;
    struct TNode* next;
}Node;

Node* head;
Node* tail;

Node* makeNode(int v){
    Node* p = (Node*)malloc(sizeof(Node));
    p->value = v; p->next = NULL;
    return p;
}
```

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI – CODE HOÀN CHỈNH

```
int queueEmpty(){
    return head == NULL && tail == NULL;
}
void push(int v){
    Node* p = makeNode(v);
    if(queueEmpty()){
        head = p; tail = p; return;
    }
    tail->next = p; tail = p;
}
```

```
int pop(){
    if(queueEmpty()){
        return -1;
    }
    Node* tmp = head;
    int v = head->value;
    head = head->next;
    if(head == NULL) tail = NULL;
    free(tmp);
    return v;
}
```

BÀI TẬP 2: MÔ PHỎNG HÀNG ĐỢI – CODE HOÀN CHỈNH

```
int main(){
    head = NULL; tail = NULL;
    char cmd[50];
    while(1){
        scanf("%s",cmd);
        if(strcmp(cmd,"#")==0){
            break;
        }else if(strcmp(cmd,"PUSH")==0){
            int v; scanf("%d",&v);
            push(v);
        }else if(strcmp(cmd,"POP")==0){
            if(queueEmpty()){
                printf("NULL\n");
            }else{
                int v = pop();
                printf("%d\n",v);
            }}}
    return 0;}
```

BÀI TẬP 3: KIỂM TRA NGOẶC (P.03.08.03)

- Cho một xâu chỉ chứa các kí tự ( ) [] {}. Viết một chương trình kiểm tra tính hợp lệ của xâu.

Ví dụ:

  - ([]{()}()[]): Xâu đúng
  - ([]{()]()[]): Xâu sai

- Dữ liệu:
  - Một dòng chứa một xâu (độ dài không quá 10^6)
- Kết quả:
  - Ghi 1 nếu là xâu đúng, ghi 0 nếu ngược lại.

stdin	stdout
(()[][]{}){}{}[][]({[]()})	1

BÀI TẬP 3: KIỂM TRA NGOẶC – MÃ GIẢ

- Sử dụng ngăn xếp, nếu gặp dấu mở ngoặc thì đưa vào ngăn xếp, nếu gặp đóng ngoặc thì:
  - Nếu đỉnh của ngăn xếp hiện tại là mở ngoặc tương ứng thì lấy ra khỏi ngăn xếp
  - Nếu đỉnh của ngăn xếp hiện tại không phải mở ngoặc tương ứng thì là xâu sai.

```
match(a,b){
if (a=='(' and b ==')') or (a=='{' and
b=='}')
or (a=='[' and b == ']') return true;
return false;
}
```

```
Check(s){
stack h;
for i = 1…len(s){
   if s[i] in ('(', '[', '{') then push(s[i]);
   else if match(s[i],top(h)) then pop(h);
   else return false;
}
if h is empty return true;
return false;
}
```

BÀI TẬP 3: KIỂM TRA NGOẶC – CODE HOÀN CHỈNH

```
const int N = 1e6;
char s[N];

bool match(char a, char b){
    if(a == '(' && b == ')') return true;
    if(a == '[' && b == ']') return true;
    if(a == '{' && b == '}') return true;
    return false;
}
```

```
bool check(char* s){
    //cout << "len(s) = " << strlen(s) << endl;
    stack<char> S;
    for(int i= 0; i < strlen(s); i++){
        if(s[i] == '(' || s[i] == '{' || s[i] == '['){
            S.push(s[i]);
            //cout << "Push s[" << i << "] = " << s[i] << endl;
        }else{
            if(S.empty()) return false;
            char a = S.top(); S.pop();
            //cout << "POP a = " << a << endl;
            if(!match(a,s[i])) return false;
        }
    }
    //cout << S.size() << endl;
    return S.empty();
}
```

BÀI TẬP 4: NHỮNG BÌNH NƯỚC (P.03.08.04)

- Có 2 bình nước dung tích lần lượt là a lít và b lít (a,b là hai số nguyên). Có một cái hồ nước với lượng nước vô hạn. Cho một số nguyên dương c, làm thế nào để có được chính xác c lít nước?
- Dữ liệu:
  - Line 1: Chứa 3 số nguyên dương a,b,c ($1 \leq a, b, c \leq 900$).
- Kết quả:
  - Ghi ra số bước tối thiểu cần làm hoặc ghi ra -1 nếu không có giải pháp.

Ví dụ:

stdin	stdout
6 8 4	4

BÀI TẬP 4: NHỮNG BÌNH NƯỚC – MÃ GIẢ

- Liệt kê các bước chuyển đổi trạng thái về lượng nước của hai bình (cặp hai số nguyên x,y) bằng hàng đợi để có số bước chuyển đổi ngắn nhất.
  - Đánh dấu trạng thái nếu cặp (x,y) đã được duyệt qua
  - Với mỗi trạng thái (x,y) ở đầu hàng đợi thì thêm vào cuối hàng đợi những trạng thái chưa được duyệt nhưng có thể tiến tới từ (x,y) bằng 1 bước. Số bước được sử dụng tăng thêm 1.
  - Nếu gặp trạng thái đích thì trả về số bước, nếu kết thúc duyệt mà không gặp thì trả về -1

```
mark(t){
    m[t] = 1;
}
```

```
is_mark(t){
    return m[t] == 1;
}
```

```
target(t){
    return c in t;
}
```

```
update_num_steps(t,r){
    num[t] = r + 1;
}
```

```
next_steps(t){
    r = list_of_next_1_step_from(t);
    return r;
}
```

```
Check(a,b,c){
(x,y) = (0,0); mark((0,0));
queue q; q.push((x,y))
while not empty(q){
t = pop(q);
for ti in next_steps(t){
   if target(ti) then return num[t]+1];
   if not is_mark(ti) then {push(ti); mark(ti);
update_num_steps(ti, num[t]);}
}
}
return -1;
}
```

BÀI TẬP 4: NHỮNG BÌNH NƯỚC – CODE HOÀN CHỈNH

```
void init(){
    for(int i = 0; i < MAX; i++)
        for(int j = 0; j < MAX; j++)
            visited[i][j] = 0;
}
int goal(int x, int y){
    return x == c || y == c;
}
```

```
int fillJug2(int x, int y){
    if(goal(x,b)){ ans = level[x][y] + 1; return 1;}
    if(visited[x][b]) return 0;
    qx.push_back(x);
    qy.push_back(b);
    visited[x][b] = 1;
    level[x][b] = level[x][y] + 1;
    return 0;
}
```

```
int fillJug1(int x, int y){
    if(goal(a,y)){ ans = level[x][y] + 1; return 1;}
    if(visited[a][y]) return 0;
    qx.push_back(a);
    qy.push_back(y);
    visited[a][y] = 1;
    level[a][y] = level[x][y] + 1;
    return 0;
}
```

BÀI TẬP 4: NHỮNG BÌNH NƯỚC – CODE HOÀN CHỈNH

```
int emptyJug1(int x, int y){
    if(goal(0,y)){
        ans = level[x][y] + 1; return 1;
    }
    if(visited[0][y]) return 0;
    qx.push_back(0);
    qy.push_back(y);
    visited[0][y] = 1;
    level[0][y] = level[x][y] + 1;
    return 0;
}
```

```
int emptyJug2(int x, int y){
    if(goal(x,0)){
        ans = level[x][y] + 1; return 1;
    }
    if(visited[x][0]) return 0;
    qx.push_back(x);
    qy.push_back(0);
    visited[x][0] = 1;
    level[x][0] = level[x][y] + 1;
    return 0;
}
```

BÀI TẬP 4: NHỮNG BÌNH NƯỚC – CODE HOÀN CHỈNH

```
int pourJug1ToJug2(int x, int y){
    int nx, ny;
    if(x + y > b){
        nx = x+y-b; ny = b;
    }else{
        nx = 0; ny = x+y;
    }
    if(goal(nx,ny)){ ans = level[x][y] + 1;return
1;}
    if(visited[nx][ny]) return 0;
    qx.push_back(nx);
    qy.push_back(ny);
    visited[nx][ny] = 1;
    level[nx][ny] = level[x][y] + 1;
    return 0;
}
```

```
int pourJug2ToJug1(int x, int y){
    int nx, ny;
    if(x + y > a){
        nx = a; ny = x+y-a;
    }else{
        nx = x+y; ny = 0;
    }
    if(goal(nx,ny)){ ans = level[x][y] + 1;return
1;}
    if(visited[nx][ny]) return 0;
    qx.push_back(nx);
    qy.push_back(ny);
    visited[nx][ny] = 1;
    level[nx][ny] = level[x][y] + 1;
    return 0;
}
```

BÀI TẬP 4: NHỮNG BÌNH NƯỚC – CODE HOÀN CHỈNH

```
void solve(){
    init();
    qx.push_back(0);
    qy.push_back(0);
    level[0][0] = 0;
    visited[0][0] = 1;
    ans = -1;
    while(!qx.empty()){
        int x = qx.front(); qx.pop_front();
        int y = qy.front(); qy.pop_front();
        //cout << "pop(" << r << "," << c << ")" << endl;

        if(fillJug1(x,y)) break;
        if(fillJug2(x,y)) break;
        if(emptyJug1(x,y)) break;
        if(emptyJug2(x,y)) break;
        if(pourJug1ToJug2(x,y)) break;
        if(pourJug2ToJug1(x,y)) break;}
   printf("%d",ans);}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !