REPORT 24120184_WEEK5

STACK:

1.Hàm Stack* initializeStack();

- Định nghĩa Stack mới, ta cấp phát bộ nhớ cho stack mới
- Cho top = nullptr (stack ban đầu rỗng)
- Trả về con trỏ trỏ tới Stack.

2.Hàm void push(Stack* &s, int key); (thêm vào đỉnh của Stack)

-Viết hàm CreateNode(int data);

- Khai báo 1 node mới bằng hàm CreateNode với kiểu dữ liệu data hay key.
- Trỏ p_next của node vừa mới tạo tới top.
- Cập nhật top bằng node mới (để node mới nằm ở đỉnh của danh sách liên kết).

3. Hàm pop(Stack* &s); (lấy ra phần tử đỉnh của Stack)

-Lấy giá trị và xóa phần tử ở đỉnh Stack

-Trường hợp nếu Stack rỗng (top == nullptr) trả về -1.

-Trường hợp còn lại:

- Khai báo 1 biến save kiểu dữ liệu int để lưu giá trị của node đầu tiên.
- Cập nhật top bằng node kế tiếp.
- Xóa node cũ và trả về save (key của node xóa).

4.Hàm int size(Stack* s);

-Đếm số lượng Node có trong Stack.

- Khai báo biến cnt = 0.
- Khai báo biến cur bằng top, duyệt qua từ top xuống cuối mỗi node đi qua tăng biến cnt lên 1.
- Trả về cnt.

5.bool isEmpty(Stack* s);

-Kiểm tra Stack có rỗng hay không.

Nếu top bằng nullptr thì Stack rỗng trả về true ngược lại trả về false.

1 số hàm hỗ trợ khác: deleteStack(Stack* &s) , printStack(ofstream &out, Stack *s)

Hàm deleteStack:

-Nếu Stack rỗng thì thoát khỏi hàm

-Nếu Stack không rỗng:

- Khai báo node cur bằng top duyệt từ đỉnh xuống , qua mỗi node khai báo node temp di chuyển node cur đến node tiếp theo và xóa node temp.
- Cho top bằng nullptr, xóa stack.

Hàm PrintStack(ofstream &out,Stack *s):

-Nếu Stack rỗng in vào file output EMPTY.

-Nếu Stack không rỗng:

Khai báo node cur bằng top duyệt qua từng node và in ra giá trị của từng node.

QUEUE:

1.Hàm Queue* initializeQueue();

- Định nghĩa Queue mới, ta cấp phát bộ nhớ cho Queue mới
- Cho head bằng nullptr và tail bằng nullptr (Queue ban đầu rỗng)
- Trả về con trỏ trỏ tới Queue mới tạo.

2.Hàm void enqueue(Queue* &q, int key); (thêm vào cuối Queue)

-Viết hàm CreateNode(int data)

-Thêm phần tử mới vào cuối hàng đợi(theo nguyên tắc FIFO)

- Khai báo 1 node mới bằng hàm CreateNode với kiểu dữ liệu data hay key.
- Nếu Queue rỗng (head == nullptr) , gán cho head = tail = newnode.
- Nếu không rỗng, nối tail->p_next bằng newnode, rồi cập nhật lại vị trí tail bằng newnode.

3. Hàm dequeue(Queue* &q); (lấy ra phần tử đỉnh của Queue)

-Lấy giá trị và xóa phần tử ở đầu của Queue.

-Trường hợp nếu Queue rỗng (head == nullptr) trả về -1.

-Khai báo 1 biến save kiểu dữ liệu int để lưu giá trị của node đầu tiên.

-Trường hợp nếu chỉ có 1 node (head->p_next == nullptr), gán 1 node temp bằng head, cập nhật cả head và tail về nullptr, xóa node temp và trả về giá trị của key của node vừa xóa.(save)

-Trường hợp còn lại:

- Khai báo node temp bằng head.

- Cập nhật con trỏ head đến node kế tiếp
- Xóa node temp và trả về save (key của node xóa).

4.Hàm int size(Queue* q);

-Đếm số lượng Node có trong queue.

- Khai báo biến cnt = 0.
- Khai báo biến cur bằng hea, duyệt qua từ head tới tail mỗi node đi qua tăng biến cnt lên 1.
- Trả về cnt.

5.bool isEmpty(Queue* q);

- Kiểm tra Queue có rỗng hay không.
- Nếu head bằng nullptr thì Queue rỗng trả về true ngược lại trả về false.

2 hàm hỗ trợ còn lại deleteQueue(Queue* &q) và printQueue(ofstream &out, Queue* q):

Cách thức ghi hàm tương tự như ở Stack.

Ảnh up code lên github: