TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM

Báo cáo bài tập 05

- Phạm Quang Sang
- MSSV: 24120429

17/04/2025

- Stack
  -
```
typedef struct Node
{
    int key;
    Node *next = nullptr;
} Node;
```
```
typedef struct Stack
{
    Node *top = nullptr;
} Stack;
```
  - Initialize a Stack
    -
```
Stack *initializeStack()
{
    Stack *s = new Stack;
    return s;
}
```
    - Ý tưởng: cấp phát động cho con trỏ s kiểu Struct* bằng toán tử new.
  - Push a new element onto the Stack
    -
```
void push(Stack **s, int key)
{
    if(*s==nullptr) *s = initializeStack();
    Node *n = createNode(key);
    n->next = (*s)->top;
    (*s)->top = n;
}
```
```
Node *createNode(int key)
{
    Node *new_node = new Node;
    new_node->key = key;
    return new_node;
}
```
    - Ý tưởng:
      - Hàm createNode(): cấp phát động cho con trỏ new_node kiểu Node* bằng toán tử new. Gán thành viên key của Node new_node = key được truyền vào.
      - Hàm push(): tạo con trỏ Node* n, gán nó vào con trỏ Node* được tạo bằng hàm createNode với thành viên key được gán bằng key được truyền vào. Nối next của node n bằng node top stack, trỏ node top stack vào n.
      - Nếu s chưa được khởi tạo, gọi hàm initializeStack() để khởi tạo và tiếp tục push.
  - Pop the top element off the Stack and return its value

```
int pop(Stack *s)
{
    if(s==nullptr||s->top==nullptr) return -1;
    else
    {
        int top_val = s->top->key;
        Node *temp = s->top;
        s->top = s->top->next;
        delete temp;
        return top_val;
    }
}
```

- Ý tưởng:
  - Nếu stack chưa được khởi tạo hoặc stack rỗng: trả về -1
  - Ngược lại, lưu key và địa chỉ node top stack vào các biến tạm, trỏ top sang next của top, xóa bộ nhớ và trả về key của node top cũ.

- Return the number of elements in the Stack

```
int size(Stack *s)
{
    if(s==nullptr) return -1;

    Node*cur = s->top;
    int count = 0;
    while (cur != nullptr)
    {
        count++;
        cur = cur->next;
    }
    return count;
}
```

- Ý tưởng:
  - Nếu stack chưa được khởi tạo: trả về -1
  - Ngược lại, dùng con trỏ Node* cur trỏ vào top stack và biến đếm, duyệt đến hết stack để tăng biến đếm.

- Return true if the Stack is empty

```
bool isEmpty(Stack *s)
{
    return (s==nullptr||s->top==nullptr);
}
```

- Ý tưởng:
  - Nếu stack chưa khởi tạo hoặc rỗng thì trả về true;
  - Ngược lại trả về false;

- Output

```
void showStack(Node* top, FILE *fo)
{
    if(top≠nullptr){
        showStack(top→next,fo);
        fprintf(fo,"%d ",top→key);
    }
}
void stackStatus(Stack *s,FILE*fo){
    if(isEmpty(s))
        fprintf(fo,"EMTPY\n");
    else{
        showStack(s→top, fo);
        fprintf(fo,"\n");
    }
}
```

- Ý tưởng:
  - Hàm showStack(): Nhận vào con trỏ Node* là top của stack, fo là con trỏ FILE* luồng output
    - Sử dụng đệ quy bằng cách gọi lại hàm và truyền con trỏ next của top hiện tại đến khi xuống đáy stack thì in giá trị key từng node vào luồng fo (output bài toán)
  - Hàm stackStatus():
    - Sử dụng hàm isEmpty() và showStack() để in trạng thái hiện tại của stack vào luồng fo

- Free memory

```
void freeStack(Stack **s)
{
    if(*s==nullptr) return;
    while ((*s)→top ≠ nullptr)
    {
        Node *temp = (*s)→top→next;
        delete (*s)→top;
        (*s)→top = temp;
    }
    *s = nullptr;
}
```

- Ý tưởng:
  - Nếu stack chưa được khởi tạo thì thoát hàm
  - Ngược lại:
    - Duyệt từ đỉnh xuống đáy stack, dùng con trỏ Node* tạm để lưu node tiếp theo của top, xóa bộ nhớ và gán node tiếp theo cho top
  - Trỏ stack về nullptr để tránh truy cập

không hợp lệ ở các xử lý khác

- Queue
  - ```
    typedef struct Node
    {
        int key;
        Node *next = nullptr;
    } Node;
    ```
    ```
    typedef struct Queue
    {
        Node *front = nullptr;
        Node *rear=nullptr;
    } Queue;
    ```
  - Initialize a Queue
    - ```
      Queue *initializeQueue()
      {
          Queue *s = new Queue;
          return s;
      }
      ```
    - Ý tưởng: cấp phát động cho con trỏ s kiểu Struct* bằng toán tử new.
  - Enqueue a new element into the Queue
    - ```
      void enqueue(Queue**q,int key){
          if(*q==nullptr) *q = initializeQueue();
          Node*new_node= createNode(key);
          if((*q)->front==nullptr){
              (*q)->front=(*q)->rear=new_node;
          }else{
              (*q)->rear->next = new_node;
              (*q)->rear = new_node;
          }
      }
      ```
      ```
      Node *createNode(int key)
      {
          Node *new_node = new Node;
          new_node->key = key;
          return new_node;
      }
      ```
    - Ý tưởng:
      - Nếu hàng đợi chưa được khởi tạo, khởi tạo hàng đợi bằng hàm initializeQueue() trước khi tiến hành.
      - Khởi tạo Node* new_node mới bằng hàm createNode()
        - Hàm createNode() trả về địa chỉ con trỏ Node* được cấp phát động và gán giá trị thành viên key bằng key được truyền vào
      - Nếu hàng đợi đang trống thì trỏ front và rear vào new_node
      - Ngược lại, nối new_node vào cuối hàng đợi và trỏ rear vào new_node
  - Dequeue the front element from the Queue and return its value
      ```

```
int dequeue(Queue*q){
    if(q==nullptr||q->front==nullptr)
        return -1;
    else{
        int val = q->front->key;
        Node*temp = q->front;
        if(q->front==q->rear)
            q->front = q->rear = nullptr;
        else
            q->front = q->front->next;
        delete temp;
        return val;
    }
}
```

- Ý tưởng
  - Nếu hàng đợi chưa được khởi tạo hoặc đang trống, trả về -1
  - Ngược lại:
    - Lưu key và địa chỉ node* front vào các biến tạm có kiểu tương ứng.
    - Nếu hàng đợi chỉ có 1 phần tử, trỏ front và rear cùng về nullptr
    - Ngược lại
      - Trỏ front sang node kế tiếp
    - Xóa bộ nhớ node front cũ và trả về giá trị key

- Return the number of elements in the Queue

```
int size(Queue *q){
    if(q==nullptr) return -1;
    int count = 0;
    Node*cur = q->front;
    while(cur!=nullptr){
        count++;
        cur=cur->next;
    }
    return count;
}
```

- Ý tưởng
  - Nếu hàng đợi chưa được khởi tạo, trả về -1
  - Ngược lại, duyệt từ đầu đến cuối hàng đợi, tăng biến đếm
  - Kết thúc vòng lặp, biến đếm có giá trị bằng số lượng phần tử. Trả về biến đếm count

- Return true if the Queue is empty

```
bool isEmpty(Queue *q){
    return (q==nullptr||q->front==nullptr);
}
```

- Ý tưởng:
  - Nếu q chưa được khởi tạo hoặc đang trống thì trả về true

        - Ngược lại, trả về false
    - Output

```
void queueStatus(Queue *q,FILE*fo){
    if(isEmpty(q))
        fprintf(fo,"EMPTY");
    else{
        Node*cur = q→front;
        while(cur≠nullptr){
            fprintf(fo,"%d ",cur→key);
            cur = cur→next;
        }
    }
    fprintf(fo,"\n");
}
```

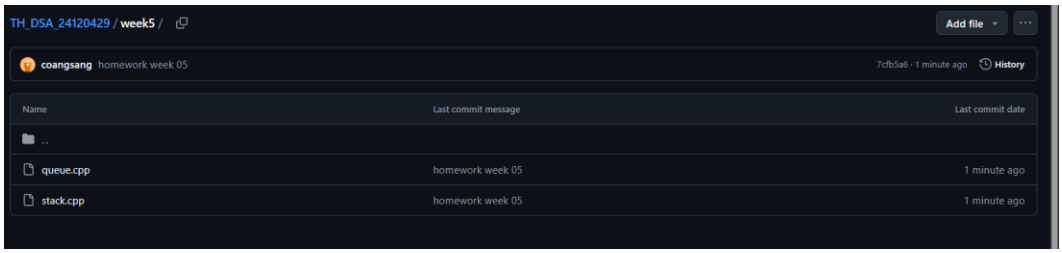
        - Luồng output: fo
        - Ý tưởng
            - Nếu q chưa được khởi tạo hoặc đang rỗng, in ra EMPTY
            - Ngược lại, duyệt từ đầu đến cuối hàng đợi, in ra key từng phần tử
            - Kết thúc bằng ký tự xuống dòng
    - Free memory

```
void freeQueue(Queue**q){
    if(*q==nullptr) return;
    while((*q)→front≠nullptr){
        Node*temp = (*q)→front→next;
        delete (*q)→front;
        (*q)→front = temp;
    }
    *q = nullptr;
}
```

        - Ý tưởng
            - Nếu hàng đợi chưa được khởi tạo: thoát hàm
            - Ngược lại:
                - Duyệt từ đầu đến cuối, dùng dùng Node* tạm lưu địa chỉ node kế tiếp, xóa bộ nhớ front hiện tại, trỏ front sang node kế tiếp qua node*tạm
                - Trỏ hàng đợi về nullptr để tránh truy cập bộ nhớ không hợp lệ

➢ GIT

```
PS D:\0HOCTAP\TH_DSA_24120429> git add week5/queue.cpp
PS D:\0HOCTAP\TH_DSA_24120429> git add week5/stack.cpp
warning: in the working copy of 'week5/stack.cpp', LF will be replaced by CRLF the next time Git touches it
PS D:\0HOCTAP\TH_DSA_24120429> git commit -m "homework week 05"
[main 7cfb5a6] homework week 05
 2 files changed, 281 insertions(+)
 create mode 100644 week5/queue.cpp
 create mode 100644 week5/stack.cpp
PS D:\0HOCTAP\TH_DSA_24120429> git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 20 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 3.30 KiB | 3.30 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/coangsang/TH_DSA_24120429
   a65e62d..7cfb5a6  main -> main
PS D:\0HOCTAP\TH_DSA_24120429>
```

- 
TH_DSA_24120429 / week5 /
Add file
coangsang homework week 05
7cfb5a6 · 1 minute ago
History
Name
Last commit message
Last commit date
..
queue.cpp
homework week 05
1 minute ago
stack.cpp
homework week 05
1 minute ago