Cấu trúc dữ liệu - Queue

uit.edu.vn

Ngày 27 tháng 3 năm 2023

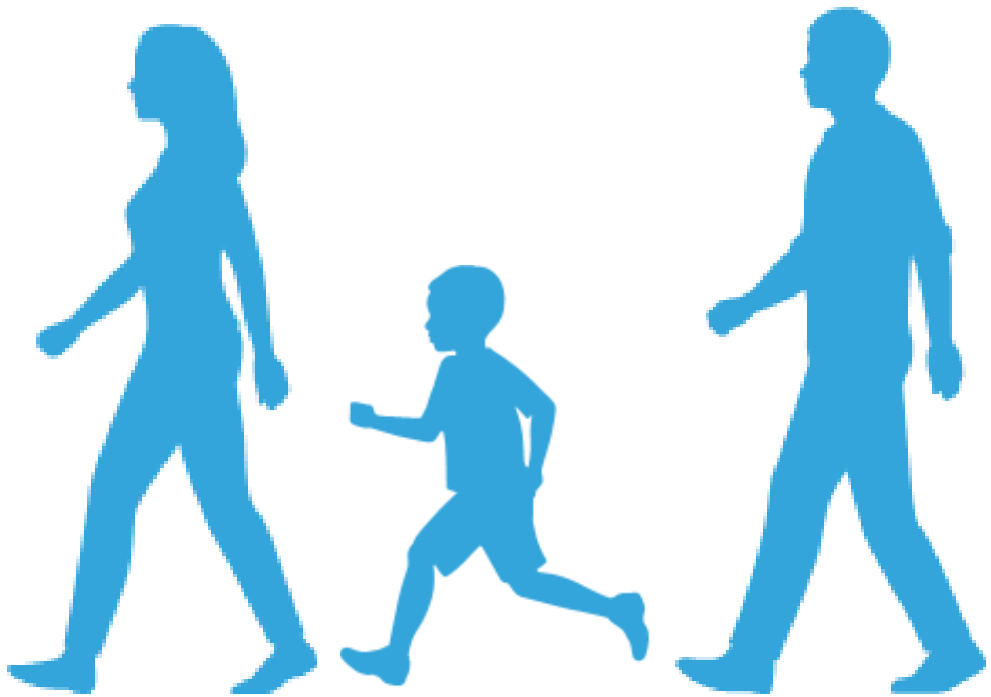
Minh hoạ Queue

Hình 1: FIFO: First In First Out: Vào trước ra trước

Minh hoạ Queue

Hình 1: FIFO: First In First Out: Vào trước ra trước

Minh hoạ Queue

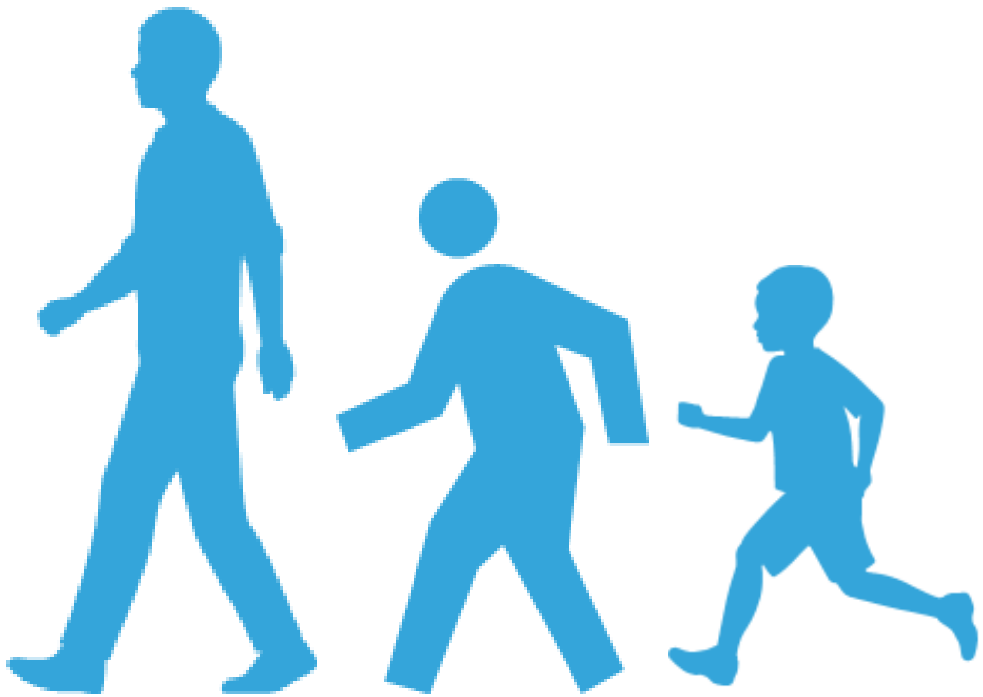


Hình 1: FIFO: First In First Out: Vào trước ra trước

Minh hoạ Queue

Hình 1: FIFO: First In First Out: Vào trước ra trước

Minh hoạ Queue



Hình 1: FIFO: First In First Out: Vào trước ra trước

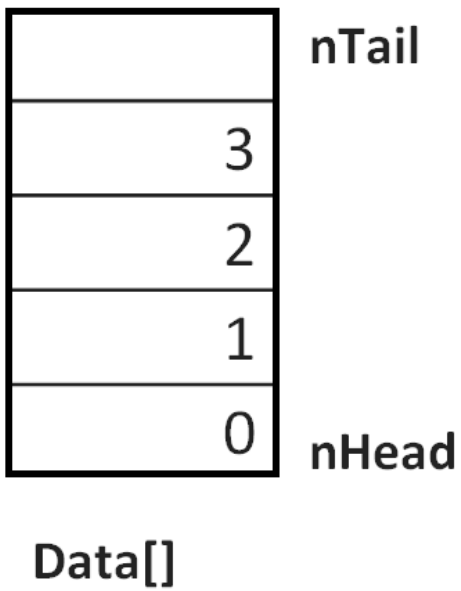
Minh hoạ Queue

Hình 1: FIFO: First In First Out: Vào trước ra trước

Cài đặt lớp Queue

Khai báo 1 lớp Queue có các hàm sau:

- empty(): kiểm tra hàng đợi có rỗng không? (test whether container is empty)
- size(): lấy kích thước của hàng đợi (return size)
- push(): thêm 1 phần tử vào hàng đợi (insert element)
- pop(): Xoá 1 phần tử ra khỏi hàng đợi (remove next element)
- front(): Chỉ đến phần tử đầu của hàng đợi (access next element)
- bạck(): Chỉ đến phần tử cuối của hàng đợi (access last element)

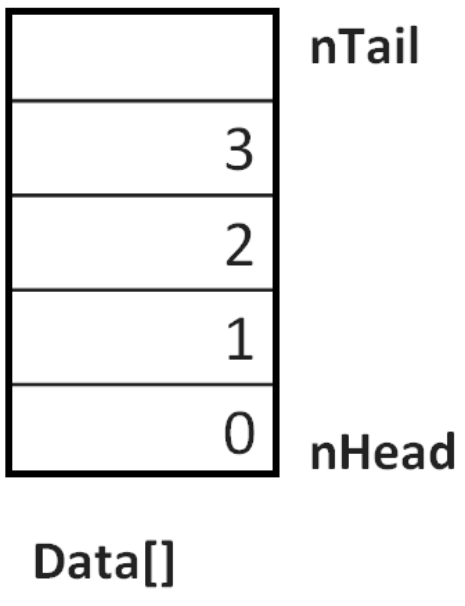


Cài đặt lớp Queue

Khai báo 1 lớp Queue có các hàm sau:

- empty(): kiểm tra hàng đợi có rỗng không? (test whether container is empty)
- size(): lấy kích thước của hàng đợi (return size)
- push(): thêm 1 phần tử vào hàng đợi (insert element)
- pop(): Xoá 1 phần tử ra khỏi hàng đợi (remove next element)
- front(): Chỉ đến phần tử đầu của hàng đợi (access next element)
- bạck(): Chỉ đến phần tử cuối của hàng đợi (access last element)

```
class Queue
{
        int Data[MAX]
        int nHead,nTail;
}
```



Bài tập: Cài đặt lớp queue

```
1  template <typename _DataType>
2  class queue{
3  private:
4      _DataType Data[MAX];
5     int nHead, nTail;
6   public: // cài đặt các hàm khởi tạo, các hàm bên dưới
7      ? empty()
8      ? push(?)
9      ? pop()
10     ? front()
11     ? back()
12 };
13 int main() {
14     queue<int> q;
15     q.push(1);      q.push(2);      q.push(3);
16     cout<<q.front();q.pop();cout<<" "<<q.empty()<<endl; //1 false
17     cout<<q.front();q.pop();cout<<" "<<q.empty()<<endl; //2 false
18     q.push(4);      q.push(5);
19     cout<<q.front();q.pop();cout<<" "<<q.empty()<<endl; //3 false
20     cout<<q.front();q.pop();cout<<" "<<q.empty()<<endl; //4 false
21     cout<<q.front();q.pop();cout<<" "<<q.empty()<<endl; //5 true
22     return 0;
23 }
```

Đáp án - Cài đặt lớp queue

```
1  template <typename _DataType>
2  class queue
3  {
4  private:
5      _DataType Data[MAX];
6      int nHead, nTail;
7  public:
8      queue(){ nHead = nTail = 0;}
9      bool empty(){return ((nTail - nHead + MAX)%MAX ==0); }
10     void push(_DataType _data)    {
11         Data[nTail] = _data;
12         nTail = (nTail+1)%MAX;
13     }
14     void pop(){              nHead = (nHead + 1)%MAX;     }
15     _DataType front()    {   return Data[(nHead)%MAX];    }
16     _DataType back()     {   return Data[(nTail-1)%MAX]; }
17
18 };
19 int main() {
20     // insert code here...
21     queue<int> q;
22     q.push(1);
```

Thay lớp queue tự viết bằng lớp queue của STL

```
1  /*template <typename _DataType>
2  class queue
3  {
4  private:
5      _DataType Data[MAX];
6      int nHead, nTail;
7  public:
8      queue(){ nHead = nTail = 0;}
9      bool empty(){return ((nTail - nHead + MAX)%MAX ==0); }
10     void push(_DataType _data)    {
11         Data[nTail] = _data;
12         nTail = (nTail+1)%MAX;
13     }
14     void pop(){              nHead = (nHead + 1)%MAX;     }
15     _DataType front()    {   return Data[(nHead)%MAX];    }
16     _DataType back()     {   return Data[(nTail-1)%MAX]; }
17
18 };*/
19 #include <queue> // dùng thư viện queue có sẵn
20 int main() {
21     queue<int> q;
22     q.push(1);
```

Priority Queue - Hàng đợi có độ ưu tiên

Cho một mảng 1 chiều, tìm giá trị lớn nhất của (nums[i]-1)*(nums[j]-1) (i != j)

Ví dụ: Input: nums = [3,4,5,2] Output: 12

Vì (4-1)*(5-1) = 3*4 = 12.

Thay lớp queue tự viết bằng lớp queue của STL

```
1    int maxProduct(vector<int>& nums) {
2            priority_queue<int> q;
3            //priority_queue<int,vector<int>,less<int>> q;
4
5            for(auto i:nums) q.push(i);
6            int max = q.top()-1;
7            q.pop();
8            max = max * (q.top()-1);
9            return max;
10       }
```

Ứng dụng

- Khử đệ qui. Ví dụ bài toán duyệt cây dùng đệ qui và stack/queue.
- Duyệt cây theo chiều rộng.
- Nên tập thói quen xếp hàng :D

Bài tập

1. Số sinh viên không thể ăn trưa. Link: LeetCode
2. Thời gian cần thiết để mua vé. Link: LeetCode
3. Tìm 2 số trong mảng có tích lớn nhất. Link: LeetCode