

QUEUE

(HÀNG ĐỢI)

23521149- Phan Đức Thành Phát

23520242 – Nguyễn Đại Trường Danh

23520133 – Phạm Phú Bảo

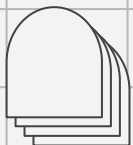
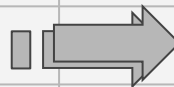
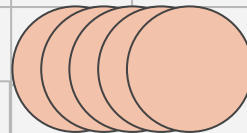


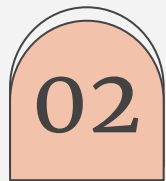


Table of contents



Khái niệm

Phương pháp cài đặt



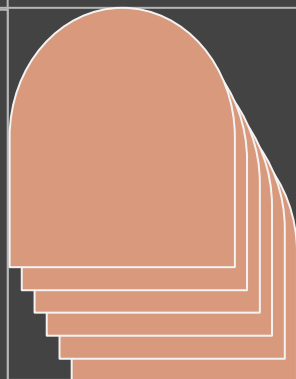
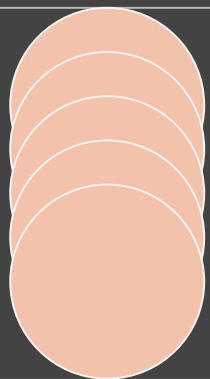
Ứng dụng

Thao tác(Operartions)



01

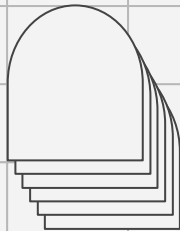
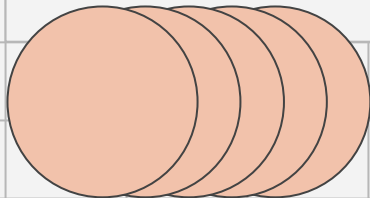
Khái niệm



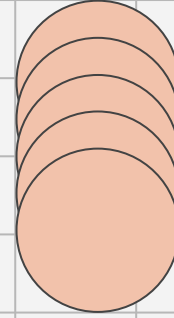


Định nghĩa:

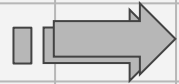
Hàng đợi (tiếng Anh: Queue) là một cấu trúc dữ liệu trừu tượng dùng để lưu trữ các đối tượng theo cơ chế **FIFO** (First In First Out) - **vào trước ra trước**.



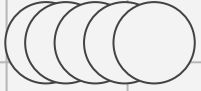
Đặc điểm



- **Mở cả 2 đầu:** một đầu dùng để thêm dữ liệu(enqueue), đầu còn lại để lấy dữ liệu(dequeue)
- **Cơ chế FIFO:** Đối tượng được thêm vào đầu tiên sẽ là đối tượng được lấy ra đầu tiên.
- **Có thể truy cập cả hai đầu:** Tuy nhiên, chỉ có thể lấy dữ liệu từ đầu ra.



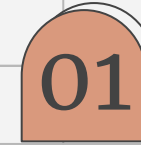
Discussion



- Tổ chức hàng đợi theo nghĩa thông thường.



Hàng đợi vòng – Circular Queues



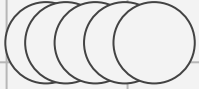
Hàng đợi tuyến tính -Linear Queues



Hàng đợi ưu tiên – Priority Queues

DSA

Discussion



- Tổ chức hàng đợi theo nghĩa thông thường.
- Giải quyết việc thiếu bộ nhớ khi sử dụng hàng đợi.

01

Hàng đợi tuyến tính -Linear Queues

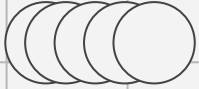
02

Hàng đợi vòng – Circular Queues

03

Hàng đợi ưu tiên – Priority Queues

Discussion



01

Hàng đợi tuyến tính - Linear Queues

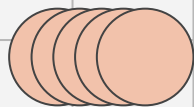
03

Hàng đợi ưu tiên – Priority Queues

02

Hàng đợi vòng – Circular Queues

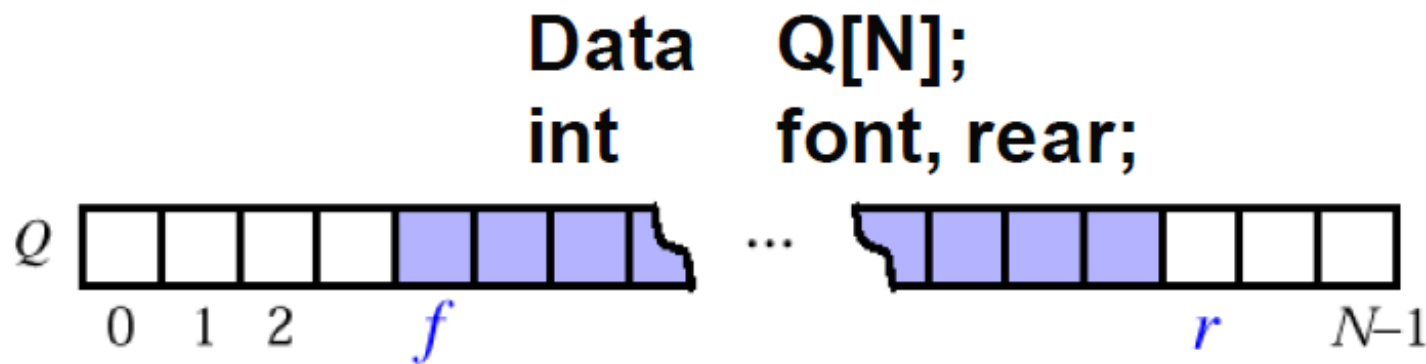
- Tổ chức hàng đợi theo nghĩa thông thường.
- Giải quyết việc thiếu bộ nhớ khi sử dụng hàng đợi.
- Mỗi phần tử có kết hợp thêm thông tin về độ ưu tiên.
- Khi chương trình cần lấy một phần tử khỏi hàng đợi, nó sẽ xét những phần tử có độ ưu tiên cao trước.



Cách cài đặt



- Dùng mảng 1 chiều




Cài đặt Queue bằng mảng 1 chiều




- *Cấu trúc dữ liệu:*

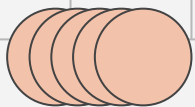
```
typedef struct tagQueue {  
    int a[MAX];  
    int Front; //chỉ số của phần tử đầu trong Queue  
    int Rear; //chỉ số của phần tử cuối trong Queue  
} Queue;
```

- *Khởi tạo Queue rỗng*



```
void CreateQueue(Queue &q) {  
    q.Front = -1;  
    q.Rear = -1;  
}
```





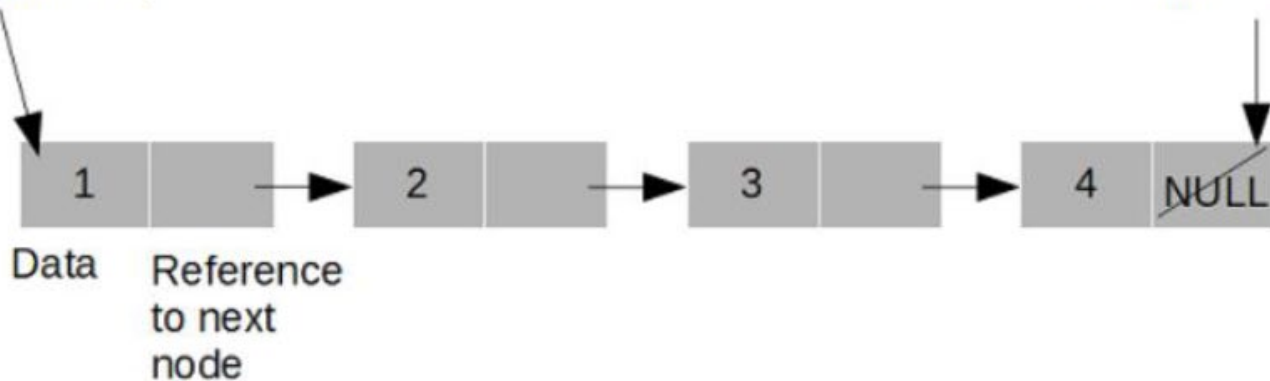
Cách cài đặt



- Dữ liệu danh sách liên kết đơn

Front (Element removed from this end)

Rear (Element added to this end)



Cài đặt Queue DSLK



Kiểm tra Queue có rỗng?

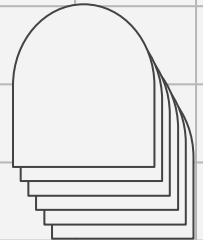
// Cấu trúc của một node

```
struct NODE {  
    int info;  
    NODE* pNext;  
};
```

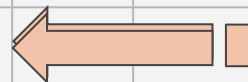
// Cấu trúc của một DSLK

```
struct QUEUE {  
    NODE* front;  
    NODE* back;  
};
```

```
bool isEmpty(QUEUE &Q) {  
    if (Q.front == NULL) //Queue rỗng  
        return 1;  
    return 0;  
}
```



Basic Operations



1. EnQueue(O): Thêm đối tượng O vào cuối hàng đợi.
2. DeQueue(): Lấy đối tượng ở đầu hàng đợi
3. isEmpty(): Kiểm tra xem hàng đợi có rỗng hay không?
4. Front(): Trả về giá trị của phần tử nằm đầu hàng đợi mà không hủy nó.

EnQueue(O): Thêm đối tượng O vào cuối hàng đợi.

Queues

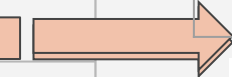


1

2

3

Enqueue()



```
void queueEnqueue(int data)
{
    // Check queue is full or not
    if (capacity == rear) {
        printf("\nQueue is full\n");
        return;
    }

    // Insert element at the rear
    else {
        queue[rear] = data;
        rear++;
    }
    return;
}
```

DeQueue(): Lấy đối tượng ở đầu hàng đợi

Queues

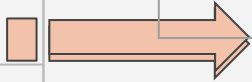


1

2

3

Dequeues()



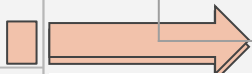
```
void queueDequeue()
{
    // If queue is empty
    if (front == rear) {
        printf("\nQueue is empty\n");
        return;
    }

    // Shift all the elements from index 2
    // till rear to the left by one
    else {
        for (int i = 0; i < rear - 1; i++) {
            queue[i] = queue[i + 1];
        }

        // decrement rear
        rear--;
    }
    return;
}
```

isEmpty(): Kiểm tra xem hàng đợi có rỗng hay không?

Queues



```
// This function will check whether
// the queue is empty or not:
bool isEmpty()
{
    if (front == -1)
        return true;
    else
        return false;
}
```


Front(): Trả về giá trị của phần tử nằm đầu hàng đợi mà không hủy nó.

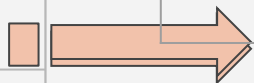
Queues



1

2

3



```
// Function to get front of queue
int front(Queue* queue)
{
    if (isempty(queue))
        return INT_MIN;
    return queue->arr[queue->front];
}
```

Queue trong STL



Cú pháp khởi tạo

Khai báo thư viện: `#include<queue>`

Khai báo Queue: `queue < kiểu dữ liệu > name;`

Ví dụ: `Queue <int> Q;`



Các Hàm với Queue

push

front

empty

pop

size




Các hàm với queue

```
1  #include<iostream>
2  #include<queue>
3  using namespace std;
4  int main(){
5      queue<int> a;
6      queue<string> b;
7      for(int i = 1; i <= 5; i++){
8          a.push(i);
9      }
10     cout << "So phan tu cua a la " << a.size() << endl;
11     while(!a.empty()){
12         cout << a.front() << endl;
13         a.pop();
14     }
```


```
15     cout << "-----" << endl;
16     b.push("abc");
17     b.push("ab");
18     b.push("a");
19     while(!b.empty()){
20         cout << b.front() << " ";
21         b.pop();
22     }
23 }
```



Ứng dụng



Tổ chức lưu vết các quá trình tìm kiếm theo chiều rộng, và quay lui vết cận



Tổ chức quản lý và phân phối tiến trình trong các hệ điều hành.



Tổ chức bộ đệm bàn phím



Bài toán minh họa tìm kiếm theo chiều rộng:

EXAMPLE

Cho hai số nguyên dương a và b và hai thao tác dưới đây:

- **Thao tác 1:** Trừ a đi 1 đơn vị
- **Thao tác 2:** Nhân b lên 2

Tìm số thao tác tối thiểu để biến đổi a thành b ?



```

1  #include<iostream>
2  #include<queue>
3  using namespace std;
4  int Cnt[1000000];
5  void Solve(int s, int t){
6      queue<pair<int,int>> q;
7      q.push({s,0});
8      while(!q.empty()){
9          pair<int,int> x = q.front();
10         // kiểm tra xem đã xuất hiện t chưa
11         // nếu đã xuất hiện thì in ra số bước nhỏ nhất
12         q.pop();
13         if(x.first == t){
14             cout << x.second;
15             return;
16         }

```

```

17         // Thao tác 1 (trừ 1)
18         if(x.first > 1 && Cnt[x.first - 1] == 0){
19             q.push({x.first-1,x.second+1});
20             Cnt[x.first - 1] = 1;
21         }
22         // Thao tác 2 (nhân đôi)
23         if(x.first < t && Cnt[x.first * 2] == 0){
24             q.push({x.first*2, x.second + 1});
25             Cnt[x.first * 2];
26         }
27     }
28 }
29 using namespace std;
30 int main(){
31     int start;
32     int end;
33     cin >> start >> end;
34     Solve(start, end);
35 }

```

