Bài: Queue và Deque

Xem bài học trên website để ủng hộ Kteam: Queue và Deque

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Bên cạnh stack thì queue cũng là một cấu trúc dữ liệu hết sức thông dụng. Ngoài ra, một biến thể được coi là sự kết hợp của stack và queue là deque được ứng dụng phổ biến trong các bài toán. Trong bài học ngày hôm nay, hãy cùng nhau đi tìm hiểu về queue và deque để xem nó là gì nhé!

Nội dung

Để có thể hiểu được bài học này một cách tốt nhất, các bạn nên có kiến thức cơ bản về các phần:

- Biến, kiểu dữ liệu, toán tử trong C++
- Câu điều kiện, vòng lặp, hàm trong C++
- Stack trong C++
- Các kiến thức cần thiết để theo dõi khóa học
- Nhập, xuất dữ liệu qua file trong C++
- Và đừng quên Cài đặt môi trường CodeBlocks để thực hành theo hướng dẫn

Trong bài học ngày hôm nay, chúng ta sẽ tìm hiểu về:

- Khái niệm queue và cách sử dụng
- Khái niệm deque và cách sử dụng

Queue

Khái niệm

Nếu như stack là một cấu trúc dữ liệu dạng "vào sau, ra trước" (Last In First Out) thì queue là một cấu trúc dữ liệu dạng "vào trước, ra trước" (First In First Out), có nghĩa là phần tử nào được vào trong queue trước sẽ được ra trước.

Một ví dụ minh hoạ thực tế nằm chính ở tên gọi tiếng Việt của queue là hàng đợi. Queue giống như một hàng người xếp hàng mua vé vậy, người đến sau sẽ vào cuối hàng, người đến trước được mua vé trước và sau khi mua vé xong sẽ đi ra khỏi hàng để đến người tiếp theo.

Một queue sẽ hỗ trợ các thao tác sau:

- Thêm phần tử vào cuối queue
- Loại bỏ phần tử ở đầu queue
- Lấy ra phần tử đầu tiên trong queue
- Lấy ra kích thước của queue

Sử dụng queue trong C++

Trong khoá học này, mình sẽ không giới thiệu cách cài đặt queue thủ công do việc này sẽ khó khăn hơn so với stack và gần như là không có ứng dụng trong thực tiễn.

Khai báo queue

Thông thường để thêm queue vào chương trình, chúng ta sẽ thêm thư viện như sau:

```
#include<queue>
```

Tuy nhiên, ở trong suốt khoá học này mình sẽ sử dụng header sau:

#include<bits/stdc++.h>

Header này sẽ giúp chúng ta thêm tất cả các thư viện về các cấu trúc dữ liệu mà chúng ta sẽ học trong khoá học này.

Ta sẽ khai báo queue như sau:

queue <{kiểu dữ liệu}> {tên queue};

Ví dụ: queue<int> myQueue;

Các phương thức cơ bản của queue

Queue trong C++ sẽ hỗ trợ các phương thức sau:

- **push**: Thêm phần tử vào cuối queue
- **pop**: Loại bỏ phần tử ở đầu queue
- **front**: Trả về giá trị là phần tử đầu tiên trong queue
- **size**: Trả về số nguyên là kích thước của queue
- **empty**: Trả về giá trị bool, true nếu queue rỗng, false nếu queue không rỗng

Các phương thức trên đều mất độ phức tạp O(1).

> Lưu ý: Cũng như stack, các phương thức pop và front khi được gọi phải đảm bảo queue không rỗng nếu không sẽ gây ra Runtime Error. Do đó, nếu không chắc chắn, các bạn sẽ cần kiểm tra bằng phương thức empty trước khi gọi hai phương thức này.

Ở đây mình có một đoạn code demo các phương thức cơ bản của queue:

C++:

```
#include<bits/stdc++.h>
using namespace std;

queue<int> q;

int main(){
    // Thêm các phần tử vào queue
    q.push(1);
    q.push(3);
    q.push(5);
    // Lúc này, queue là [1, 3, 5]

    // In ra phần tử đầu tiên trong queue
    cout << "Phan tu dau tien trong queue la: " << q.front() << endl;
    // In ra kích thước của queue
    cout << "Kich thuoc cua queue la: " << q.size() << endl;

    // Loại bỏ phần tử đầu tiên ra khỏi queue
    cout << "Loai bo phan tu dau trong queue" << endl;
    q.pop();
    // Khi này queue là [3, 5]

    // Kiểm tra queue có rỗng hay không
    if(q.empty()) cout << "Queue rong" << endl;
    else cout << "Queue khong rong" << endl;

    // In ra phần tử đầu tiên trong queue
    cout << "Phan tu dau tien trong queue la: " << q.front() << endl;
    // In ra kích thước của queue
    cout << "Kich thuoc cua queue la: " << q.size() << endl;
}
```

Khi chạy đoạn code trên, ta thu được kết quả:

```
Phan tu dau tien trong queue la: 1
Kich thuoc cua queue la: 3
Loai bo phan tu dau trong queue
Queue khong rong
Phan tu dau tien trong queue la: 3
Kich thuoc cua queue la: 2
```

Deque

Khái niệm

Trong bài học trước, mình đã giới thiệu về stack, một cấu trúc dữ liệu cho phép thêm dữ liệu ở cuối và lấy ra dữ liệu ở cuối. Vừa rồi, mình đã giới thiệu thêm cho các bạn về queue, một cấu trúc dữ liệu cho phép thêm dữ liệu ở cuối và lấy ra dữ liệu ở đầu. Vậy thì có cấu trúc dữ liệu nào có thể kết hợp các tính chất của stack và queue hay không? Câu trả lời chính là deque.

Deque là viết tắt của *double-ended queue*, có nghĩa là hàng đợi hai đầu. Một deque sẽ hỗ trợ các phươ ng thức sau:

- Thêm một phần tử vào cuối deque
- Thêm một phần tử vào đầu deque
- Bỏ đi phần tử ở cuối deque
- Bỏ đi phần tử ở đầu deque
- Lấy ra giá trị phần tử đầu deque
- Lấy ra giá trị phần tử cuối deque

Sử dụng deque trong C++

Giống như với queue, việc cài đặt deque thủ công là tương đối phức tạp và không cần thiết nên mình sẽ không hướng dẫn các bạn cài đặt deque thủ công.

Khai báo deque

Thông thường để thêm deque vào chương trình, chúng ta sẽ thêm thư viện như sau:

#include<deque>

Tuy nhiên, ở trong suốt khoá học này mình sẽ sử dụng header sau:

#include<bits/stdc++.h>

Header này sẽ giúp chúng ta thêm tất cả các thư viện về các cấu trúc dữ liệu mà chúng ta sẽ học trong khoá học này.

Ta sẽ khai báo deque như sau:

deque <{kiểu dữ liệu}> {tên deque};

Ví dụ: deque<int> myDeque;

Các phương thức cơ bản của deque

Deque trong C++ sẽ hỗ trợ các phương thức cơ bản sau:

- **push_front**: Thêm phần tử vào đầu deque
- **push_back**: Thêm phần tử vào cuối deque
- **pop_front**: Loại bỏ phần tử ở đầu deque
- **pop_back**: Loại bỏ phần tử ở cuối deque
- **front**: Trả về giá trị là phần tử đầu trong deque
- **back**: Trả về giá trị là phần tử cuối trong deque
- **size**: Trả về giá trị nguyên là kích thước của deque
- **empty**: Trả về giá trị bool, true nếu deque rỗng, false nếu deque không rỗng

Các phương thức trên đều mất độ phức tạp O(1).

> Lưu ý: Cũng như các cấu trúc dữ liệu khác, các phương thức pop_front, pop_back, front, back khi được gọi phải đảm bảo deque không rỗng nếu không sẽ gây ra Runtime Error. Do đó, nếu không chắc chắn, các bạn sẽ cần kiểm tra bằng phương thức empty trước khi gọi các phương thức trên.

Ở đây mình có một đoạn code demo về các phương thức của deque như sau:

C++:

```
#include<bits/stdc++.h>
using namespace std;

deque<int> dq;

int main(){
    // Thêm 3, 6 vào cuối deque
    dq.push_back(3);
    dq.push_back(6);
    // Lúc này deque là [3, 6]
    // Thêm 4 vào đầu deque
    dq.push_front(4);
    // Lúc này deque là [4, 3, 6]
    // Thêm 1 vào đầu deque
    dq.push_front(1);
    // Lúc này deque là [1, 4, 3, 6]

    cout << "Kich thuoc cua deque la: " << dq.size() << endl;
    cout << "Phan tu dau tien trong deque la: " << dq.front() << endl;
    cout << "Phan tu cuoi cung trong deque la: " << dq.back() << endl;

    cout << "Xoa bo phan tu cuoi deque" << endl;
    dq.pop_back();
    // Lúc này deque là [1, 4, 3]

    cout << "Kich thuoc cua deque la: " << dq.size() << endl;
    cout << "Phan tu dau tien trong deque la: " << dq.front() << endl;
    cout << "Phan tu cuoi cung trong deque la: " << dq.back() << endl;

    cout << "Xoa bo phan tu dau deque" << endl;
    dq.pop_front();
    // Lúc này deque là [4, 3]

    cout << "Kich thuoc cua deque la: " << dq.size() << endl;
    cout << "Phan tu dau tien trong deque la: " << dq.front() << endl;
    cout << "Phan tu cuoi cung trong deque la: " << dq.back() << endl;
}
```

Khi chạy đoạn code trên ta thu được kết quả sau:

```
Kich thuoc cua deque la: 4
Phan tu dau tien trong deque la: 1
Phan tu cuoi cung trong deque la: 6
Xoa bo phan tu cuoi deque
Kich thuoc cua deque la: 3
Phan tu dau tien trong deque la: 1
Phan tu cuoi cung trong deque la: 3
Xoa bo phan tu dau deque
Kich thuoc cua deque la: 2
Phan tu dau tien trong deque la: 4
Phan tu cuoi cung trong deque la: 3
```

Ứng dụng trong thực tế của queue và deque

Khi học bài này, sẽ có bạn thấy lạ: Tại sao mình không nêu ra một bài toán ban đầu rồi suy luận để đi đến cấu trúc dữ liệu như trong bài trước? Lí do là vì queue và deque hầu như không đứng độc lập để giải quyết các bài toán.

Queue được ứng dụng trong các thuật toán thiên về yếu tố cần duyệt và lưu trữ các trạng thái, điển hình cho dạng thuật toán như vậy là loang và BFS. Đối với deque, thuật toán ứng dụng quan trọng nhất là tìm kiếm min-max trên đoạn tịnh tiến. Các thuật toán trên sẽ đều có trong khoá học này của mình. Khi đi đến thuật toán đó, mình sẽ hướng dẫn cho các bạn cách sử dụng cụ thể.

Tất nhiên, queue và deque sẽ có thể ứng dụng trong các bài toán khác. Tuỳ vào tính chất của bài toán và tính chất của queue và deque, các bạn có thể sử dụng chúng linh hoạt sao cho phù hợp.

Kết luận

Qua bài này chúng ta đã nắm được về queue và deque.

Bài sau chúng ta sẽ bắt đầu tìm hiểu về cấu trúc dữ liệu **Linked List**.

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".