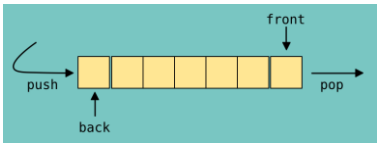
BÀI GIẢNG VỀ KỸ THUẬT LẬP TRÌNH VỚI C++

04 CẤU TRÚC DỮ LIỆU HÀNG ĐỢI - QUEUE

A. LÝ THUYẾT

1. Khái niệm

Hàng đợi – Queue là một danh sách các phần tử cùng kiểu mà phép bổ sung (**push**) một phần tử được thực hiện tại một đầu và phép lấy ra (**pop**) một phần tử được thực hiện tại đầu còn của danh sách.



Đầu được bổ sung gọi là phía sau (*back*), đầu được lấy ra gọi là phía trước (*front*). Các Các phần tử được đưa vào trước sẽ được lấy ra trước, vì vậy ta còn nói việc bổ sung hay lấy ra các phần tử theo quy tắc *First In – First Out* (*vào trước ra trước*).

- Cấu trúc hàng đợi gợi cho ta hình ảnh xếp hàng vào mua vé.

- Các chương trình trong máy tính được hệ điều hành quản lý về thứ tự thực hiện theo hàng đợi có hạn chế thời gian.

Khai báo

Ngôn ngữ lập trình C^{++} hỗ trợ kiểu dữ liệu **queue** và được khai báo như sau.

queue <*kiểu phần tử*> <*tên biến queue* >;

Ví dụ: **queue** <*int*> myQueue;

Khai báo một hàng đợi mà kiểu các phần tử của nó là kiểu ***int***.

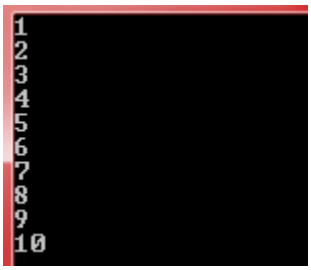
2. Các phương thức trên kiểu dữ liệu stack

Phương thức	Thực hiện
*myQueue.**empty***()	Trả về giá trị ***true*** nếu hàng đợi rỗng, ngược lại trả về ***false***.
my*Queue.**push(x)***	Đưa phần tử x vào hàng đợi.
*myQueue.**front***()	Trả về giá trị của phần đầu (front) hàng đợi – xem hình vẽ
*myQueue.**back*** ()	Trả về giá trị của phần sau (back) hàng đợi – xem hình vẽ.
*myQueue.**pop***()	Lấy phần tử ở đầu (front) ra khỏi hàng đợi.
*myQueue.**size***()	Trả về số phần tử hiện có trong hàng đợi.

Ví dụ 1:

```
#include<bits/stdc++.h>
using namespace std;
queue<int>  myQueue;
int main(){
  for(int i = 1; i <= 10; ++i)
       myQueue.push(i);
  while(!myQueue.empty()){
    cout<<myQueue.front()<<'\n';
    myQueue.pop();
  }
}
```

Kết quả:



Ví dụ 2: Digits tree and queue

Từ ba chữ số 3, 5, 7; hãy tạo ra tất cả các số có 1 chữ số và có 2 chữ số.

→ myQueue.**push**({3, 5, 7});

3	5	7							

x = myQueue.**front**(); myQueue.**pop**()
→ x = 3

	5	7							

myQueue.**push**({33, 35, 37})

	5	7	33	35	37				

x = myQueue.**front**(); myQueue.**pop**()
→ x = 5

		7	33	35	37				

myQueue.**push**({53, 55, 57})

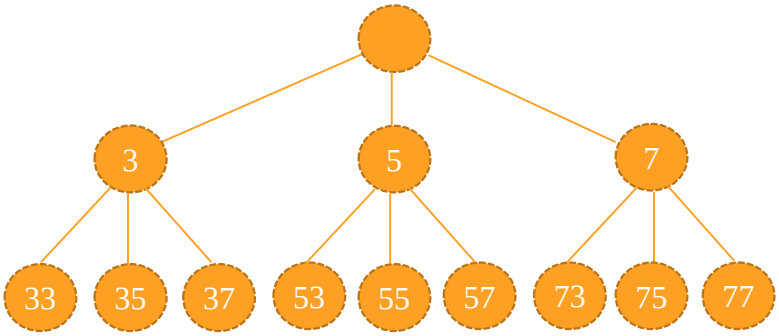
		7	33	35	37	53	55	57				

x = myQueue.**front**(); myQueue.**pop**()
→ x = 7

			33	35	37	53	55	57				

myQueue.**push**({73, 75, 77})

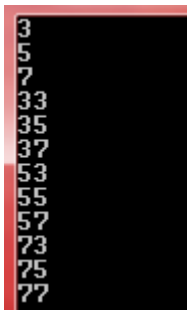
			33	35	37	53	55	57	73	75	77	



Như vậy, các số lần lượt được đưa và hàng đợi tạo thành dãy số cần tạo.

```cpp
#include<bits/stdc++.h>
using namespace std;
queue<int>  myQueue;
int main(){
  myQueue.push(3);
  myQueue.push(5);
  myQueue.push(7);
  int x;
  while (!myQueue.empty()){
    x = myQueue.front();
    myQueue.pop();
    cout<<x<<'\n';
    if(x*10+3 <100) myQueue.push(x*10+3); //x*10 + 3 la so co 2 chu so
    if(x*10+5 <100) myQueue.push(x*10+5); //x*10 + 5 la so co 2 chu so
    if(x*10+7 <100) myQueue.push(x*10+7); //x*10 + 7 la so co 2 chu so
  }
  }
```

Kết quả:



B. BÀI TẬP

1☼ Tạo số

Cho k chữ số khác nhau và khác 0 ($1 \leq k \leq 9$). Gọi S là tập các số được tạo bởi k chữ số đã cho. Sắp xếp dãy số này theo thứ tự tăng dần.

Yêu cầu: Đưa ra n số thứ n của dãy sắp xếp trên.

Dữ liệu cho trong file TAOSO.INP gồm:

- Dòng đầu ghi hai số k và n ($n \leq 10^6$).
- Dòng hai ghi k chữ số khác nhau và khác 0.

Kết quả ghi ra file TAOSO.OUT là số thứ n tìm được.

Ví dụ:

TAOSO.INP	TAOSO.OUT
3 7 3 5 7	53

2☼ Ghép chữ số

Cho k chữ số $a_1, a_2, \ldots, a_k$ ($k \leq 10$) và số nguyên dương M. Hỏi có số nguyên E mà chỉ gồm các chữ số trong k chữ số đã cho, số chữ số của E ít hơn 8 chữ số và chia hết cho M hay không? Nếu có thì đưa ra số E nhỏ nhất.

Ví dụ: Cho các chữ số: 1, 2, 7, 8 và $M = 6$, số E tìm được là 12.

Cho các chữ số: 1, 2, 7, 8 và $M = 100$. Khi đó không có số E nào thỏa mãn bài toán.

Dữ liệu cho trong file GHEPSO.INP như sau:

- Dòng đầu ghi số nguyên dương k là số các chữ số ($1 \leq k \leq 10$) và số nguyên dương M.
- Dòng thứ hai ghi k chữ số.

Kết quả ghi ra file GHEPSO.OUT là số nguyên E nhỏ nhất tìm được, nếu không tồn tại số nguyên dương E thì ghi ra số -1.

Ví dụ:

GHEPSO.INP	GHEPSO.OUT
4 88 1 7 9 2	792
4 8 1 9 2 6	16
4 66666666 1 2 4 6	-1

3☼ Dịch chuyển trên lưới

Cho lưới ô vuông gồm m dòng và n cột. Các dòng được đánh số từ 1 đến m (từ trên xuống dưới), các cột được đánh số từ 1 đến n (từ trái sang phải). Ô ở dòng i cột j được gọi là ô (i, j). Trên một số ô có chướng ngại vật. Một con robot được đặt tại ô (1, 1) cần di chuyển đến (m, n). Tại mỗi bước đi, từ một ô có thể di chuyển sang bốn ô kề cạnh với điều kiện:

- Không di chuyển ra ngoài lưới.
- Ô không có chướng ngại vật.

Yêu cầu: Tìm đường di chuyển của robot đi từ ô (1, 1) đến ô (m, n) với số ô đi qua ít nhất.

Dữ liệu cho trong file văn bản SHORTEST.INP gồm:

- Dòng đầu ghi hai số nguyên dương m, n ($n,m \leq 10^3$).
- m dòng tiếp theo ghi n số nguyên 0 hoặc 1 mô tả, 1 là ô có chướng ngại vật, 0 là ô không có chướng ngại vật.

Kết quả ghi ra file SHORTEST.OUT là số ô ít nhất cần đi qua để đi từ ô (1, 1) đến ô (m, n), tính cả ô (1, 1) và (m, n). Nếu không có cách di chuyển thì ghi -1.

Ví dụ:

SHORTEST.INP	SHORTEST.OUT
5 4 0 0 0 0 1 1 0 1 1 0 0 1 0 0 1 1 0 0 0 0	10
3 4 0 0 0 1 0 1 1 0 1 0 0 0	-1

4☼. Swap_divisor

Cho hai số nguyên dương n và k. Số k được gọi là ***swap_divisor*** của số n nếu ta có thể thực hiện: Từ số n biến đổi thành n_1, từ n_1 biến đổi thành n_2, ..., cứ thực hiện như vậy để nhận được số m và số m chia hết cho k. Với mỗi lần biến đổi, ta tráo đổi hai chữ số kề nhau để được một số mới.

Ví dụ: n = 1032, k = 10, ta có k là ***swap_divisor*** của n, vì n = 1032 → 1302 → 1320 chia hết cho k = 10.

Với $n = 111, k = 12$, ta có k không phải là ***swap_divisor*** của n.

Chú ý là nếu n chia hết cho k thì ta cũng nói k ***swap_divisor*** của n (khi đó không cần biến đổi).

Yêu cầu: Cho n và k, kiểm tra xem k có phải là ***swap_divisor*** của n hay không?

Dữ liệu cho trong file SWAPDIVISOR.INP gồm:

- Dòng đầu ghi số nguyên dương Q ($Q \leq 10$) là số lần kiểm tra.
- Q dòng tiếp theo, mỗi dòng ghi hai số nguyên dương n và k.

Kết quả ghi ra file SWAPDIVISOR.OUT gồm Q dòng, mỗi dòng ghi 1 nếu trên dòng tương ứng, k là ***swap_divisor*** của n.

Ví dụ:

SWAPDIVISOR.INP	SWAPDIVISOR.OUT
3 12 2 321 3 1113 2	1 1 0

Giới hạn:

- 50% số test ứng với $n, k \leq 10^7$.
- 50% số test khác ứng với $n, k \leq 10^9$.

5☼. Táo thối

Siêu thị Apples supermarket có một gian hàng trưng bày táo rất lớn. Khu hàng được chia làm m dòng và n cột. Ô ở dòng i và cột j được gọi là ô (i, j). Tại mỗi ô (i, j) có thể đặt một quả táo hoặc không có quả táo nào. Do số lượng táo nhiều và bán chưa hết nên có những quả táo bị hỏng, có những quả táo chưa hỏng. Cứ sau một ngày, nếu một quả táo chưa bị hỏng thì sẽ bị hỏng nếu các ô bên cạnh có một quả táo đã bị hỏng (có nhiều nhất 4 ô bên cạnh). Cho biết hiện trạng các quả táo bị hỏng và chưa bị hỏng, hỏi bao nhiêu ngày nữa thì tất quả táo có trong siêu thị bị hỏng.

Dữ liệu cho trong file ROTA.INP như sau:

- Dòng đầu ghi hai số nguyên dương m và n.
- m dòng tiếp theo, mỗi dòng ghi n số thuộc {0, 1, 2} mô tả trạng thái của n quả táo trên dòng đó: số 0 mô tả ô không có táo, 1 mô tả ô chứa quả táo chưa bị hỏng, 2 mô tả ô chứa quả táo bị hỏng.

Kết quả ghi ra file ROTA.OUT là số ngày ít nhất để tất cả các quả táo trong siêu thị đều bị hỏng. Nếu không có thì ghi ra -1.

Ví dụ:

ROTA.INP	ROTA.OUT
3 5 2 1 0 2 1 1 0 1 2 1 1 0 0 2 1	2
3 5 2 1 0 2 1 0 0 1 2 1 1 0 0 2 1	-1

Giới hạn: $n, m \leq 1000$.

6☼. Địa đạo

Trong chiến tranh, người ta xây dựng rất nhiều địa đạo. Địa đạo XYZ là một địa đạo khổng lồ, được cấu thành từ các đường hầm tạo thành một lưới ô vuông gồm 1002 đường hầm dọc, 1002 đường hầm ngang. Các đường hầm dọc được đánh số từ 0 đến 1001 theo hướng từ trái sang phải, các đường hầm ngang được đánh số 0 đến 1001 theo hướng từ trên xuống dưới. Ở tại mỗi điểm giao của đường hầm dọc và đường hầm ngang đều có một cánh cửa. Cánh cửa giao giữa đường hầm ngang i và đường hầm dọc j thì ta nói cặp (i, j) là tọa độ của cánh cửa. Những cánh cửa này giúp địa đạo được an toàn hơn, hiện địa đạo có k cánh cửa đang đóng, các cánh của còn lại đều mở. Ban chỉ huy đang đứng ở cửa có tọa độ (x, y), cửa này đang mở và muốn di chuyển tới cửa có tọa độ (0, 0) theo các đường hầm. Hãy tính xem, Ban chỉ huy cần mở ít nhất bao nhiêu cánh cửa đển có thể đi tới được cánh cửa có tọa độ (0; 0), cửa tọa độ (0; 0) luôn được mở.

Dữ liệu cho trong file DIADAO.INP như sau:

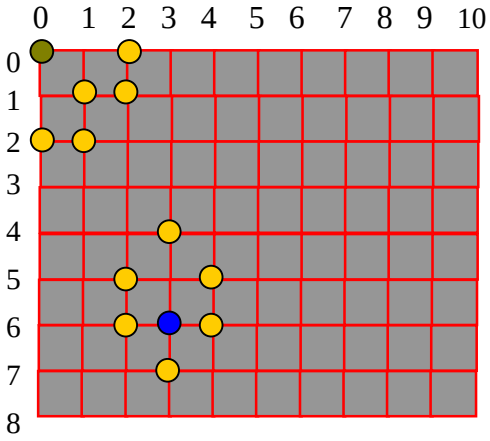
- Dòng 1: Gồm ba số nguyên dương cách nhau là k, x, y là số lượng cánh cửa đang đóng và tọa độ của Ban chỉ huy.
- Dòng 2 đến dòng k+1: Mỗi dòng chứa hai số (x, y) là tọa độ của cánh cửa đang đóng.

Kết quả ghi ra file DIADAO.OUT là số cửa ít nhất cần mở.

Ví dụ:

DIADAO.INP	DIADAO.OUT
11 6 3 6 2 5 2 4 3 2 1 7 3 5 4 6 4 2 0 1 1 1 2 0 2	2

Giới hạn: $1 \leq k \leq 50.000$


0 1 2 3 4 5 6 7 8 9 10
0
1
2
3
4
5
6
7
8