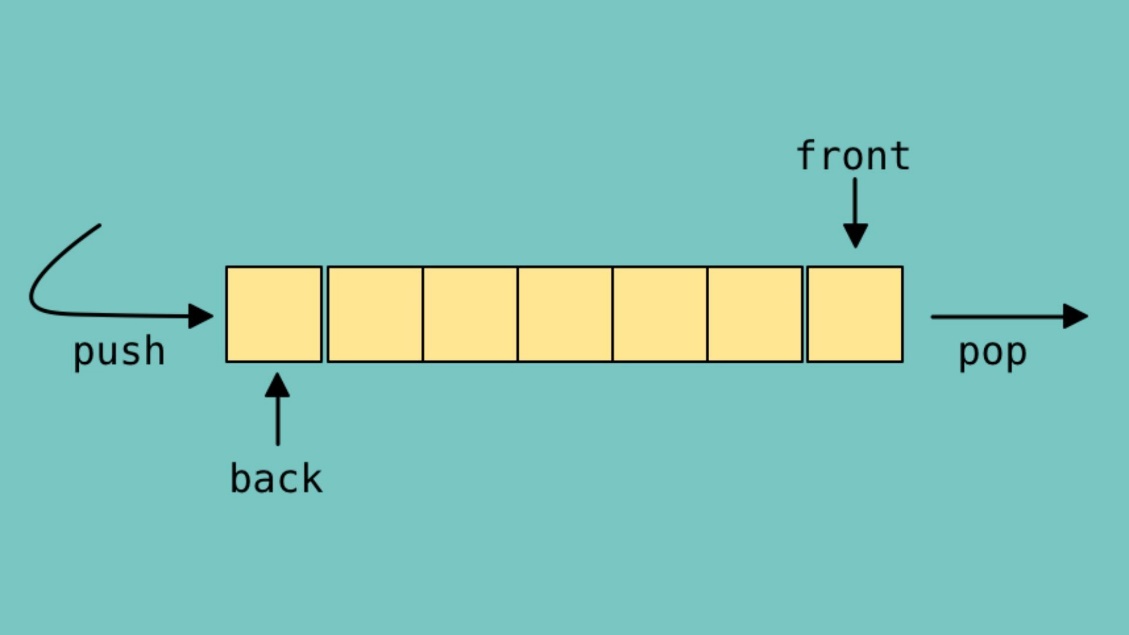
* Hàng đợi ưu tiên (Priority Queue) trong C++ là một cấu trúc dữ liệu được sử dụng để lưu trữ các phần tử theo một thứ tự ưu tiên nhất định.
* Bản chất của queue :



* Với QUEUE thì thằng nào vào trước thì sẽ được trích ra trước (Thằng nào vào xếp hàng thanh toán BIG C trước thì sẽ được thu ngân thanh toán trước ).
* Tuy nhiên trong hàng đợi ưu tiên thì lại khác, các phần tử được trích xuất ra dựa trên mức độ ưu tiên của chúng, với phần tử có mức độ ưu tiên cao nhất sẽ được rút trích ra trước.
* Cấu trúc dữ liệu này thường được triển khai bằng cách sử

dụng heap, một cấu trúc dữ liệu nhị phân đặc biệt

* Các thao tác chính :

+ push () : Thêm một phần tử vào hàng đợi ưu tiên.

+ pop () : Xóa phần tử có mức độ ưu tiên cao nhất ra khỏi hàng đợi ưu tiên.

+ top () : Truy xuất phần tử có mức độ ưu tiên cao nhất mà không xóa nó khỏi hàng đợi.

#include <iostream>

#include <queue>

int main() {

priority\_queue<int> pq;

// Thêm các phần tử vào hàng đợi ưu tiên

pq.push(30);

pq.push(10);

pq.push(50);

pq.push(20);

// In ra các phần tử từ hàng đợi ưu tiên

while (!pq.empty()) {

cout << pq.top() << " "; // In phần tử có mức độ ưu tiên cao nhất

pq.pop(); // Xóa phần tử đó khỏi hàng đợi ưu tiên

}

return 0;

}

* Kết quả : **50 30 20 10**
* Ta có thể thấy rõ được các phần tử sau khi được đưa vào hàng đợi ưu tiên đã được sắp xếp theo thứ tự giảm dần của sự ưu tiên ( Giá trị nào lớn hơn sẽ được ưu tiên hơn )
* Chúng ta cũng có thể đảo ngược mức độ ưu tiên của hàng đợi bằng cách rất hay dùng :

#include <iostream>

#include <queue>

#include <vector>

int main() {

priority\_queue<int,vector<int>,greater<int>> pq;

// Thêm các phần tử vào hàng đợi ưu tiên

pq.push(30);

pq.push(10);

pq.push(50);

pq.push(20);

// In ra các phần tử từ hàng đợi ưu tiên

while (!pq.empty()) {

cout << pq.top() << " "; // In phần tử có mức độ ưu tiên cao nhất

pq.pop(); // Xóa phần tử đó khỏi hàng đợi ưu tiên

}

return 0;

}

* Dùng hàng đợi ưu tiên MERGE k dãy đã sắp xếp thành một dãy đã sắp xếp duy nhất.

+ Dãy 1 : 2,5,8,12

+ Dãy 2 : 3,6,9,10

+ Dãy 3: 4,7,11,13

* Sau khi hợp nhất các dãy này sẽ là một dãy sắp xếp theo thứ tự không giảm :

2,3,4,5,6,7,8,9,10,11,12,13

* Phân tích giải pháp:

1. Bắt đầu bằng cách đưa phần tử đầu tiên của mỗi dãy vào hàng đợi ưu tiên cùng với thông tin về dãy mà phần tử đó thuộc về :

+ Cặp (2,1) – Giá trị 2 ở dãy 1

+ Cặp (3,2) – Giá trị 3 ở dãy 2

+ Cặp (4,3) – Giá trị 4 ở dãy 3

1. Lặp lại cho đến khi hàng đợi ưu tiên trống :

+ Lấy ra phần tử có mức độ ưu tiên cao nhất từ hàng đợi

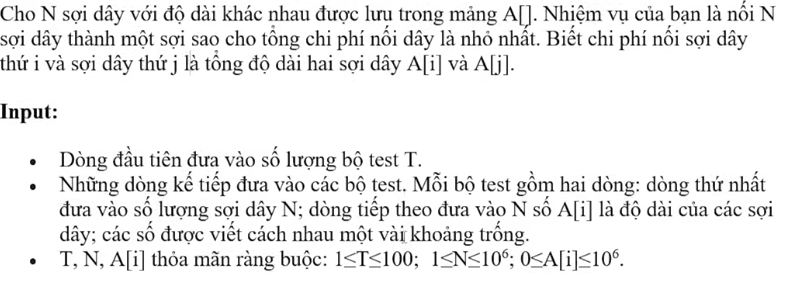
+ Thêm phần tử này vào dãy kết quả

+ Nếu dãy mà phần tử này thuộc về vẫn còn phần tử, thêm phần tử tiếp theo của dãy đó vào hàng đợi ưu tiên.

* Luôn luôn mặc định rằng cái phần tử ở top là lớn nhất , ta cũng có thể làm cho nó ngược lại bằng cách cho nó là nhỏ nhất cũng được.

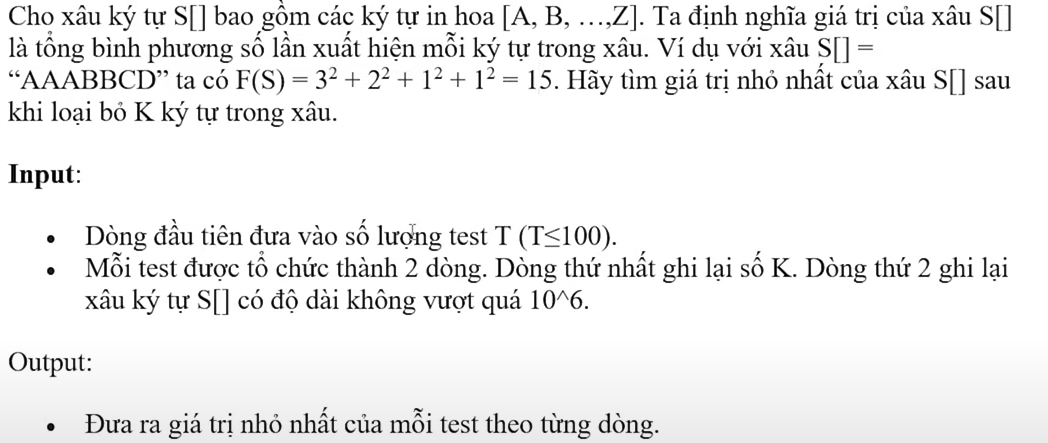
priority\_queue <int> pq;

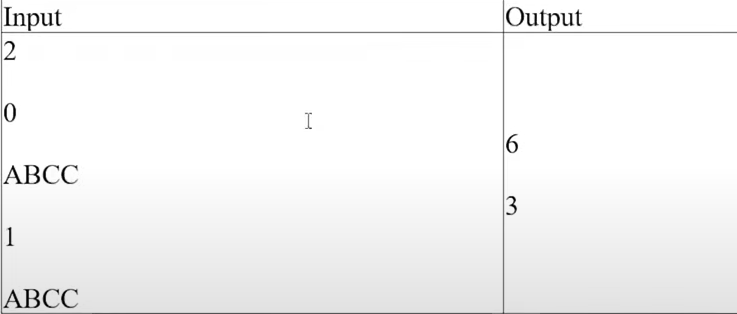
priority\_queue<int,vector<int> , greater<int>> pq;



4 3 2 6 => 29

4 2 7 6 9 => 62





- Ta hình dung rằng để có được giá trị nhỏ nhất của xâu đó thì ta phải tìm và loại bỏ đi cái thằng xuất hiện nhiều nhất.

- AAABBCD

f = 3^2 + 2^2 + 1^2 + 1^2

* Xóa đi 1 kí tự -> Xóa A : 2^2 + 2^2 + 1^2 + 1^2
* Ở mỗi bước mình sẽ chọn ra cái thằng có tần xuất lớn nhất và loại nó ra khỏi dãy.

------------------------------------------------------

* Cho một dãy số ,từ dãy số đó hãy lắp ghép nó để tạo thành một số chia hết cho 3 sao cho số này lớn nhất.

+ Một số chia hết cho 3 khi tổng các chữ số của số đó cũng chia hết cho 3.

VD : 135 % 3 vì 1+3+5 = 9 % 3 = 0

+ Một số chia hết cho 3 dư 1 khi tổng của các chữ số chia cho 3 sẽ dư 1.Số này có thể viết dưới dạng 3k+1 , với k là một số nguyên không âm

VD : 4,7,10,13,16,…. chia cho 3 sẽ dư 1

+ Một số chia cho 3 dư 2 có đặc điểm tổng các chữ số chia cho 3 sẽ dư 2 . Số này có thể viết dưới dạng 3k + 2, với k là một số nguyên không âm.

VD : 5,8,11,14,… chia cho 3 sẽ dư 2

1.a %3 = 1 để trở thành một số chia hết cho 3 thì :

+ lấy số a đó trừ đi một số chia cho 3 dư 1

+ lấy số a trừ cho hai số chia 3 dư 2 ( Tổng của hai số chia 3 dư 2 sẽ thành một số chia cho 3 dư 1 )

VD : 13 %3 = 1 -> 13-4 = 9 với 4%3=1 .

-----------------------------------------------------------

* Pair trong C++ là một kiểu container lưu trữ dữ liệu theo cặp đem lại nhiều sự tiện lợi khi bạn cần lưu trữ dữ liệu theo cặp.
* Các ví dụ có thể bắt đầu vào kết thúc của PAIR :

+ Tọa độ Oxy

+ Thời gian bắt đầu và kết thúc của một công việc.

+ Đỉnh đầu và đỉnh cuối của một cạnh trên đồ thị.

* PAIR nằm trong thư viện “utility” được sử dụng để kết hợp hai kiểu dữ liệu với nhau. Nó cung cấp cách lưu trữ 2 giá trị đi kèm nhau nhưng chỉ sử dụng 1 biến.
* Cú pháp khai báo:
* Cách 1 : Giá trị của first và second là mặc định

pair<first\_data\_type, second\_data\_type> pair\_name;

* Cách 2 : Giá trị của first là value1, second là value2

pair<first\_data\_type, second\_data\_type> pair\_name = make\_pair(value1, value2)

* Cách 3 : Giá trị của first là value1, second là value2

pair<first\_data\_type, second\_data\_type> pair\_name(value1, value2)

* Cách 4 : Giá trị của first là value1, second là value2

pair<first\_data\_type, second\_data\_type> pair\_name = {value1, value2}

-----------------------------------------------------------

#include <iostream>

#include <queue>

#include <utility> // for std::pair

#include <vector>

using namespace std;

int main() {

// Khai báo một hàng đợi ưu tiên với các cặp <int, int>

// Phần tử đầu tiên trong cặp là giá trị ưu tiên, phần tử thứ hai là dữ liệu

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;

// Thêm các phần tử vào hàng đợi ưu tiên

pq.push(make\_pair(5, 1));

pq.push(make\_pair(2, 2));

pq.push(make\_pair(8, 3));

pq.push(make\_pair(3, 4));

// Lấy các phần tử ra từ hàng đợi ưu tiên và in chúng ra màn hình

while (!pq.empty()) {

pair<int, int> current = pq.top();

pq.pop();

cout << "Priority: " << current.first << ", Value: " << current.second << endl;

}

return 0;

}

Priority: 2, Value: 2

Priority: 3, Value: 4

Priority: 5, Value: 1

Priority: 8, Value: 3

priority\_queue< pair<int, int> , vector<pair<int, int>> , greater<pair<int, int>> > pq;

+ pair <int,int> : Kiểu dữ liệu cho một phần tử trong hàng đợi ưu tiên

+ vector<pair<int,int>> : Kiểu dữ liệu của container được sử dụng để lưu trữ các phần tử trong hàng đợi ưu tiên.

+ greater<pair<int,int>> : Kiểu của hàm so sánh ưu tiên được sử dụng để so sánh ưu tiên được xác định bởi thứ tự của các phần tử trong hàng đợi ưu tiên.

* Trong trường hợp này, chúng ta sử dụng greater để sắp xếp các phần tử theo thứ tự giảm dần của các giá trị ưu tiên, tức là giá trị lớn nhất sẽ được xếp đầu tiên.

-------------------------------------------------------

* ÔN TẬP VỀ TOÁN TỬ 3 NGÔI TRONG C++ :

condition ? value\_if\_true : value\_if\_false

+ condition là một biểu thức hoặc một điều kiện logic được đánh giá.

+ value\_if\_true là giá trị được trả về nếu condition là đúng (true).

+ value\_if\_false là giá trị được trả về nếu condition là sai (false).

+ Cách hoạt động của toán tử ba ngôi rất đơn giản: nó kiểm tra điều kiện đầu vào. Nếu điều kiện là true, nó trả về giá trị được xác định cho value\_if\_true; nếu không, nó trả về giá trị được xác định cho value\_if\_false.

Dưới đây là một ví dụ minh họa về cách sử dụng toán tử ba ngôi:

int max = (x > y) ? x : y;

Trong ví dụ này, nếu x lớn hơn y, biến max sẽ được gán bằng giá trị của x; nếu không, nó sẽ được gán bằng giá trị của y. Điều này giúp ta không cần sử dụng cấu trúc if-else dài dòng để thực hiện cùng một tác vụ.

-------------------------------------------------------------

* Lời dẫn thì chúng ta khi sử dụng hàng đợi ưu tiên với pair thì nó sẽ đi kèm với việc sắp xếp theo độ ưu tiên của first trong pair nhưng khi chúng ta muốn nó ưu tiên theo second thì chúng ta phải viết biểu thức operator hoặc biểu thức lambda và từ đó đưa vào thay thế tham số “greater<pair<int,int>> “ thì nó sẽ được theo ý mình.
* Thứ nhất , ta có thể hiểu rằng cái greater<pair<int,int>> thực chất là một function (HÀM) . Điều này có thể được coi như một lớp có một hàm operator() đã được định nghĩa.

* Định nghĩa hàm so sánh tùy chỉnh để sắp xếp theo giá trị của second.

|  |
| --- |
| struct CompareSecond{  bool operator()(const pair<int,int> &left, const pair<int,int> &right) {  return left.second <right.second;  }  };  priority\_queue<pair<int, int>, vector<pair<int, int>>, CompareSecond> p;  auto cmp = [] (const pair<int,int> &left , const pair<int,int> &right) {  return left.second < right.second;  };  priority\_queue<pair<int, int>, vector<pair<int, int>>, decltype(cmp)> p(cmp); |

* Đoạn mã trên cung cấp định nghĩa một struct (Kiểu cấu trúc) có tên là ‘CompareSecond’, trong đó có một hàm toán tử gọi là

operator(). Hàm này so sánh hai phần tử kiểu pair<int,int> dựa trên giá trị thành phần của second.

* Khi bạn khai báo priority\_queue với CompartSecond làm tham số cuối cùng, bạn đang chỉ định rằng priority\_queue nên sử dụng hàm so sánh đã được định nghĩa trong CompareSecond để sắp xếp các phần tử.
* Câu lệnh mà bạn đưa ra là một ví dụ về việc sử dụng hàng đợi ưu tiên trong trong C++

-------------------------------------------------------------------

* Đôi khi muốn lưu trữ cặp các số đi cùng nhau:

pair<datatype1, datatype2> v;

pair<int,int> p;

pair<int,int> v;

pair<string,int> p;

v. first v.second

pair<int,int> v =make\_pair(100,200);

* Đôi khi ta muốn lưu ba cặp số đi cùng nhau :

pair<pair<int,int> , int> v;

v.first.first

v.first.second

v.second

* Như vậy, ta có thể sử dụng thằng tuple :Ví dụ muốn lưu 3 số đi cùng nhau:

tuple <int,int, int> t{1,2,3}

cout<< get<0>(t) <<endl;

cout<< get<1>(t) <<endl;

cout<< get<2>(t) <<endl;

------------------------------------------------------------

* Trong C++ , tuple là một cấu trúc dữ liệu mà bạn có thể sử dụng để lưu trữ một tập hợp các phần tử có kiểu dữ liệu khác nhau. Cụ thể , một tuple có thể chứa một hoặc nhiều phần tử , mỗi phần tử có thể có một kiểu dữ liệu riêng.
* Tuple cung cấp một cách thuận tiện để đóng gói các giá trị không cần phải định nghĩa một cấu trúc hoặc lớp.
* Dưới đây là một số điểm cơ bản về tuple trong C++.

+ KHAI BÁO :

tuple<datatype1 , datatype2 , datatype3 > MyTuple;

+ KHỞI TẠO :

tuple<int, double, string> myTuple(10,3.14,”CHUNG”);

+ TRUY CẬP PHẦN TỬ :

int value = get<0>(myTuple); - > Truy cập phần tử đầu

double value = get<1> (myTuple); -> Truy cập phần tử thứ 2

string value = get<2> (myTuple); -> Truy cập phần tử thứ 3