優先度付きキューの話（メロリンキュー）

queue と priority_queue の違い

― FIFO と「優先度」の正体 ―

1. このページの目的

C++ STL には複数の「キュー系コンテナ」が存在します。
本ページでは、次のコードを題材に、

- `std::queue`（通常のキュー）
- `std::priority_queue`（優先度付きキュー）

の **動作原理と使いどころの違い** を整理します。

対象は **C++ が書けるゲームエンジニア科 2 年生**です。
STL の仕様寄りの説明を行います。

---

2. 使用するサンプルプログラム

2.1 プログラム全体

```
#include <iostream>
#include <queue>

namespace {
    std::priority_queue<int, std::vector<int>, std::greater<int>> myQueue;
    std::queue<int> ituQueue;
}

int main()
{
    ituQueue.push(10);
    ituQueue.push(3);
    ituQueue.push(5);
    ituQueue.push(8);
    ituQueue.push(2);

    for(int i = 0; i < 5; ++i) {
        int val = ituQueue.front();
        ituQueue.pop();
        std::cout << "Pop: " << val << std::endl;
    }

    std::cout << "-----------------------------" << std::endl;

    myQueue.push(10);
    myQueue.push(3);
    myQueue.push(5);
    myQueue.push(8);
    myQueue.push(2);

    for (int i = 0; i < 5; ++i) {
```

```
        int val = myQueue.top();
        myQueue.pop();
        std::cout << "Pop: " << val << std::endl;
    }

    return 0;
}
```

---

3. std::queue の挙動（FIFO）

3.1 std::queue とは

`std::queue` は **FIFO（First In, First Out）** のキューです。

- 先に入れたものが
- 先に出てくる

という **順番固定** のデータ構造です。

3.2 実行結果（queue）

```
Pop: 10
Pop: 3
Pop: 5
Pop: 8
Pop: 2
```

3.3 なぜこの順番になるか

```
ituQueue.push(10);
ituQueue.push(3);
ituQueue.push(5);
ituQueue.push(8);
ituQueue.push(2);
```

投入順そのままに、

```
10 → 3 → 5 → 8 → 2
```

が `front()` → `pop()` で取り出されます。

3.4 queue の用途

- イベントキュー
- 入力処理
- 通信パケット処理
- BFS（幅優先探索）

「**順番を守ること**」が最優先の処理向きです。

---

4. std::priority_queue の挙動（優先度付き）

4.1 priority_queue とは

`std::priority_queue` は、

> **「順番」ではなく「優先度」で取り出されるキュー**

です。

投入順は一切保証されません。

4.2 宣言の意味

```
std::priority_queue<int, std::vector<int>, std::greater<int>> myQueue;
```

各要素の意味は以下です。

要素	意味
`int`	格納する型
`std::vector<int>`	内部コンテナ（ヒープ用）
`std::greater<int>`	比較規約

4.3 比較規約の重要ルール

`priority_queue` では次の規約が使われます。

> **Compare(a, b) == true のとき、a は b より優先度が低い**

4.4 std::greater<int> の意味

```
std::greater<int>()(a, b)  // a > b
```

- `a > b == true`
- → **a は後回し**
- → **小さい値ほど優先**

つまり、

☒ **最小値が常に top() に来る**

4.5 実行結果（priority_queue）

```
----------------------------
Pop: 2
Pop: 3
Pop: 5
Pop: 8
Pop: 10
```

4.6 なぜこの順番になるか

投入順：

```
10, 3, 5, 8, 2
```

内部では **ヒープ構造** が作られ、

```
常に最小値が top()
```

になります。

---

5. queue と priority_queue の本質的違い

項目	std::queue	std::priority_queue
取り出し基準	入れた順	優先度
順序保証	あり	なし
内部構造	deque	heap（vector）
top / front	front()	top()

6. ゲーム開発での典型用途

6.1 std::queue

- 入力イベント処理
- メッセージキュー
- BFS（迷路探索など）

6.2 std::priority_queue

- A* / ダイクストラ法
- タスクスケジューラ
- 距離・コスト最小探索
- AI 思考順制御

7. 自作構造体での使用例（探索系）

```
struct Node {
    int cost;
};

// cost が小さいほど優先
bool operator>(const Node& a, const Node& b)
{
    return a.cost > b.cost;
}

std::priority_queue<Node, std::vector<Node>, std::greater<Node>> open;
```

```
open.push({10});
open.push({3});
open.push({7});

open.top().cost; // 3
```

8. まとめ

- `std::queue`
  - **順番重視**
  - FIFO
- `std::priority_queue`
  - **優先度重視**
  - 最小 or 最大ヒープ

探索アルゴリズムでは
「次に処理すべきもの」を選ぶために priority_queue が使われる
という点を押さえておくと、実装の意味が見えてきます。

Revision #1
★Created 30 January 2026 04:11:57 by youe2
Updated 2 June 2026 21:34:00 by youe2