# 暨南大学本科实验报告专用纸(附页)

## 基于 Heap 实现`priority_queue`

课程名称_____数据结构_____成绩评定_____
实验项目名称__基于 Heap 实现`priority_queue`__指导老师____于晓聪___
实验项目编号__15__实验项目类型__设计性__实验地点_数学系机房_
学生姓名___郭彦培___学号___2022101149___
学院_信息科学技术学院_系_数学系_专业_信息管理与信息系统_
实验时间_2024 年 6 月 13 日上午_~__2024 年 7 月 13 日中午___

## 1. 实验目的

基于 Heap 实现`priority_queue`

## 2. 实验环境

计算机：PC X64

操作系统：Windows + Ubuntu20.0LTS

编程语言：C++：GCC std20

IDE：Visual Studio Code

## 3. 程序原理

`priority_queue`要求实时维护序列中最大（或最小）的值，刚好符合堆的性质。

代码实现的`priority_queue`中提供了可选模板参数`typename Compare = std::less<VALUE_TYPE>`，若传入`std::greater`则为最大值优先，反之为最小值优先。

# 4. 程序代码

## 4.1. `priority_queue.h`

```cpp
#ifndef PRIORITY_QUEUE_HPP
#define PRIORITY_QUEUE_HPP

#include <vector>
#include <functional>

#ifdef __PRIVATE_DEBUGE
#include <iostream>
#endif

namespace myDS {
    template<typename VALUE_TYPE, typename Compare = std::less<VALUE_TYPE>>
    class priority_queue {
    private:
        std::vector<VALUE_TYPE> h;
        Compare comp;

        void floow(std::size_t x) {
            while (x > 1 && comp(h[x / 2], h[x])) {
                std::swap(h[x], h[x / 2]);
                x >>= 1;
            }
        }

        void drown(std::size_t x) {
            while (x * 2 <= h.size() - 1) {
                int t = x * 2;
                if (t + 1 <= h.size() - 1 && comp(h[t], h[t + 1])) t++;
                if (!comp(h[x], h[t])) break;
                std::swap(h[x], h[t]);
                x = t;
            }
        }

    public:
        explicit priority_queue(const Compare& comp = Compare()) : comp(comp) { h.push_back(VALUE_TYPE()); }

        ~priority_queue() { }

        void push(const VALUE_TYPE& t) {
            h.push_back(t);
            floow(h.size() - 1);
        }

        const VALUE_TYPE& top() const {
```

```
            return h[1];
        }

        VALUE_TYPE pop() {
            auto t = this->top();
            std::swap(h[1], h[h.size() - 1]);
            h.pop_back();
            drown(1);
            return t;
        }

#ifdef __PRIVATE_DEBUGE
        void innerPrint() {
            for (auto x : h) std::cout << x << " ";
            std::cout << "\n";
        }
#endif
    };
} // namespace myDS

#endif
```

### 4.2. `_PRIV_TEST.cpp`

```
#include <iostream>
#define __PRIVATE_DEBUGE
#include <Dev\15\priority_queue.h>
using namespace std;

int main()
{
     myDS::priority_queue<int,greater<int>> piq;
    while(1) {
        string s;
        cin >> s;
        if(s == "push") {
            int t;
            cin >> t;
            piq.push(t);
        } else if(s == "pop") {
            cout << piq.pop() << "\n";
        } else if(s == "top") {
            cout << piq.top() << "\n";
        } else if(s == "p") {
            piq.innerPrint();
        }
    }
}
```

## 5. 测试数据与运行结果

运行上述`_PRIV_TEST.cpp`测试代码中的正确性测试模块，得到以下内容:

```
push 5
push 4
push 1
push 3
push 2
p
0 5 4 1 3 2
pop
5
p
0 4 3 1 2
top
4
pop
4
pop
3
pop
2
pop
1
p
0
```

可以看出，代码运行结果与预期相符，可以认为代码正确性无误。