20220331-C++

1.过程描述

返回指向const对象的引用 C++ 复制代码

```C++
const Vector & Max(const & vec1,const & vec2);
1）如果返回对象，将调用复制构造函数，而返回引用则不会，因此所做的工作更少，效率更高；
2）引用所指向的对象应该在调用函数执行时已经存在
3）vec1，vec2都被声明为const引用，因此返回类型也必须为const
若方法或函数要返回局部对象，则应返回对象（使用复制构造函数来生成），而不是指向对象的引用。
```

new C++ 复制代码

```C++
String *str=new String;//将调用默认的构造函数
String *str=new String("hello");//将调用定义的构造函数如String(const char*)
String *str=new String(sayings[choice]);//将调用定义的构造函数如String(const String &);
```

一些技术 C++ 复制代码

```
1.重载<<运算符
ostream & operator<<(ostream & os,const ClassDemo & obj)
{
    os<<...;
    return os;
}

2.构造函数使用new的类
1）对于指向内存是由new分配的所有类成员，都应在类的析构函数中对其使用delete
2）如果析构函数通过对指针类成员使用delete来释放内存，则每个构造函数都应当使用new来初始化指针，或设置为空指针
3）析构函数要么使用new[]，要么使用new，不能混用
4）应定义一个分配内存（而不是将指针指向已有内存）的复制构造函数，这样程序能够将类对象初始化为另一个类对象
ClassDemo(const ClassDemo&);
5)应定义一个重载赋值运算符额类成员函数
classDemo & ClassDemo::operator=(const ClassDemo& cn)
{
    if(this==&cn)
        return *this
    delete[] ptr;
    ptr=new Class_name[size];
    ...
    return *this;
}
```

queue.h C++ 复制代码

```cpp
#pragma once
#ifndef QUEUE_H
#define QUEUE_H
class customer
{
private:
    long arrive;
    int processtime;
public:
    customer() { arrive = processtime = 0; }
    void set(long when);
    long when() const { return arrive; }
    int ptime() const { return processtime; }
};

typedef customer Item;

class queue
{
private:
    enum{Q_SIZE=10};
    struct Node
    {
        Item item;
        struct Node* next;
    };
    Node* front;
    Node* rear;
    int items;//队列中item的数量
    const int qsize;//队列中item的最大数量
    //防止public copying
    queue(const queue& q) :qsize(0){}
    queue& operator=(const queue& q) { return *this; }

public:
    queue(int qs=Q_SIZE);
    ~queue();
    bool isEmpty() const;
    bool isFull() const;
    int queuecount() const;
    bool enqueue(const Item& item);
    bool dequeue(Item& item);
};
#endif
```

queue.cpp C++ 复制代码

```cpp
#include "queue1.h"
#include <cstdlib>

queue::queue(int qs):qsize(qs)//这里用了初始化参数列表。这里由于qsize是常量，因此可以对它初始化，但不能赋值。
//对于const数据成员，必须在执行到构造函数函数体前，即创建对象时进行初始化。
//从概念上讲，调用构造函数时，对象将在括号中的代码执行前被创建
//引用与const数据类似，也只能在被创建时进行初始化。
//必须用参数化列表来初始化非静态const数据成员和引用数据成员
{
    front = rear = nullptr;
    items = 0;

}

queue::~queue()
{
    Node* temp;
    while (front != nullptr)
    {
        temp = front;
        front = front->next;
        delete temp;
    }
}

bool queue::isEmpty() const
{
    return items == 0;
}

bool queue::isFull() const
{
    return items == qsize;
}

int queue::queuecount() const
{
    return items;
}

bool queue::enqueue(const Item& item)
{
    if (isFull())
        return false;
```

```
45        Node* add = new Node;
46        add->item = item;
47        add->next = nullptr;
48        items++;
49        if (front == nullptr)
50            front = add;
51        else
52            rear->next = add;
53        rear = add;
54        return true;
55    }
56
57    bool queue::dequeue(Item& item)
58    {
59        if (front == nullptr)
60            return false;
61        item = front->item;
62        items--;
63        Node* temp = front;
64        front = front->next;
65        delete temp;
66        if (items == 0)
67            rear = nullptr;
68        return true;
69    }
70
71    void customer::set(long when)
72    {
73        processtime = std::rand() % 3 +1 ;
74        arrive = when;
75    }
```

main.cpp C++ 复制代码

```cpp
#include <iostream>
#include "queue1.h"
#include <ctime>
#include <cstdlib>

const int MIN_PER_HR = 60;
bool newcustomer(double x);

int main()
{
    using std::cin;
    using std::cout;
    using std::endl;
    using std::ios_base;
    std::srand(std::time(0));

    cout << "case study: Bank of Heather Automatic Teller\n";
    cout << "Enter maximum size of queue: ";
    int qs;
    cin >> qs;
    queue line(qs);
    cout << "Enter the number of simulation hours: ";
    int hours;
    cin >> hours;
    long cyclelimit = MIN_PER_HR * hours;
    cout << "Enter the average number of customers per hour: ";
    double perhour;
    cin >> perhour;
    double min_per_cust;
    min_per_cust = MIN_PER_HR;

    Item temp;
    long turnaways = 0;
    long customers = 0;
    long served = 0;
    long sum_line = 0;
    int wait_time = 0;
    long line_wait = 0;

    for (int cycle = 0; cycle < cyclelimit; cycle++)
    {
        if (newcustomer(min_per_cust))
        {
            if (line.isFull())
                turnaways;
```

```
            else
            {
                customers++;
                temp.set(cycle);
                line.enqueue(temp);
            }
        }
        if (wait_time <= 0 && !line.isEmpty())
        {
            line.dequeue(temp);
            wait_time = temp.ptime();
            line_wait += cycle - temp.when();
            served++;
        }
        if (wait_time > 0)
            wait_time--;
        sum_line += line.queuecount();
    }
    if (customers > 0)
    {
        cout << "customers accepted: " << customers << endl;
        cout << "  customers served: " << served << endl;
        cout << "         turnaways: " << turnaways << endl;
        cout << "average queue size: ";
        cout.precision(2);
        cout.setf(ios_base::fixed, ios_base::floatfield);
        cout << (double)sum_line / cyclelimit << endl;
        cout << " average wait time: " << (double)line_wait / served << "
minutes\n";
    }
    else
        cout << "No customers!\n";
    cout << "Done!\n";
    return 0;
}
bool newcustomer(double x)
{
    return(std::rand() * x / RAND_MAX < 1);
}
```

2.结果输出

今天没能按计划开始数据结构部分，早上跟晚上浪费了不少时间，心思没放在学习上。晚上看了一点跟入职培训相关的咨询，感觉还是挺有挑战性的，确实得收收心好好做准备。明天无论如何得开始数据结构部分。

任务项	时间	预期	碎片时间关注	实际情况（截至31日
C++ prime	3月24–26	掌握C++基本语法知识	• The Cherno c++视频 • 计算机图形学 • 游戏引擎架构 • 即兴演讲+结构化写作	还没看完，类继承以及STL两大块待完成
计算机网络	3月27–28	看完全书		未完成，还差网络安全、音频视频、两部分
计算机网络习题	3月29	利用习题检验并回顾		未完成
数据结构c语言版	3月30–4月3	看完全书		未开始，delay 了两天
数据结构与算法分析	4月4–4月10	看完全书		
TCP/IP网络编程	4月11–12	看完全书	• 数学之美 • 大图景 • 复杂性思维	
C++深度学习框架	4月13–15	看完全书		
C和指针	4月16–18	看完全书		
计算机网络自顶向下	4月19–20	看完全书		