20220327-C++&一点算法

1.过程描述

1.1 C++ Prime

运算符重载与友元函数 C++ 复制代码

运算符重载允许自定义一些运算符的运行机制。

友元函数可以访问类的私有成员，其声明放在类声明中，并在前面加上friend，在函数定义时前面不需要加friend。友元函数不是成员函数，不能使用成员运算符来调用，不能使用Time::限定符

```
----------------mytime0.h--
#pragma once
#ifndef MY_TIME_0
#define MY_TIME_0
class Time
{
private:
    int hours;
    int minutes;
public:
    Time();
    Time(int h, int m = 0);
    void AddMin(int m);
    void AddHr(int h);
    void Reset(int h = 0, int m = 0);
    Time operator+(const Time& t) const;
    Time operator-(const Time& t) const;
    Time operator*(double n) const;
    friend Time operator*(double m, const Time& t);//友元函数
    void show() const;
};
#endif
----------------mytime0.cpp--
#include "mytime0.h"
#include <iostream>

Time::Time()
{
    hours = minutes = 0;
}

Time::Time(int h, int m)
{
    hours = h;
    minutes = m;
}

void Time::AddMin(int m)
{
```

```
45      minutes += m;
46      hours += minutes / 60;
47      minutes %= 60;
48  }
49
50  void Time::AddHr(int h)
51  {
52      hours += h;
53  }
54
55  void Time::Reset(int h, int m)
56  {
57      hours = h;
58      minutes = m;
59  }
60
61  Time Time::operator+(const Time& t) const //这里返回类型不是引用，因为sum对象
    是局部变量
62  {
63      Time sum;
64      sum.minutes = minutes + t.minutes;
65      sum.hours = hours + t.hours + sum.minutes / 60;
66      sum.minutes %= 60;
67      return sum;
68  }
69
70  Time Time::operator-(const Time& t) const
71  {
72      Time diff;
73      int tot1, tot2;
74      tot1 = t.minutes + 60 * t.hours;
75      tot2 = minutes + 60 * hours;
76      diff.minutes = (tot2 - tot1) % 60;
77      diff.hours = (tot2 - tot1) / 60;
78      return diff;
79  }
80
81  Time Time::operator*(double n) const
82  {
83      Time result;
84      long totalminutes = hours * n * 60 + minutes * n;
85      result.hours = totalminutes / 60;
86      result.minutes = totalminutes % 60;
87      return result;
88  }
89
90  void Time::show() const
91  {
```

```
92          std::cout << hours << " hours, " << minutes << " minutes";
93      }
94
95      Time operator*(double m, const Time& t)
96      {
97          Time result;
98          long totalminutes = t.hours * m * 60 + t.minutes * m;
99          result.hours = totalminutes / 60;
100         result.minutes = totalminutes % 60;
101         return result;
102     }
103     ----------------main.cpp--
104     #include <iostream>
```

1.2 算法图解

算法 Python 复制代码

```
1)算法运行时间（最糟糕的情况下）分为：
1.O(n)     线性时间（例子：简单查找）
2.O(logn) 对数时间（例子：二分查找，对数的底为2）
3.O(n*logn)
4.O(n^2)
5.O(n!)
大O表示法中的n表示操作数，指出了算法运行时间的增速

2)数组与链表
数组在内存中是连续的，知道每个元素的位置。需要随机读取元素时，数组的效率比较高
链表在内存中是不连续的，前一个元素会存储它下一个元素的位置信息。如果要读取最后一个元素，则需要把所有元素访问一遍。如果需要跳跃，链表的效率比较低
常见的数组和链表的运行时间，其中O(1)为常量时间
      数组    链表
读取 O(1)    O(n)
插入 O(n)    O(1)
删除 O(n)    O(1)
链表插入元素很简单，只需要修改它前面那个元素指向的地址，而使用数组时，则需要把后面所有元素都往后移。
删除元素时，链表也是更好的选择

3)递归
递归函数包含了两个部分：
1.基线条件（什么时候不再调用自己）
2.递归条件（函数调用自己）

调用另一个函数，当前函数暂停，处于尚未完成的状态。当栈用于存储多个函数的变量，称为调用栈。
栈有两种操作：压入和弹出
```

2.结果输出

今天感觉整个人状态不在线，原本打算把C++类部分看完的，结果只粗略了看了一章，还有块硬骨头没啃。晚上看了一点算法图解，可能因为书本身比较入门，感觉看起来速度还挺快的。大致了解了算法运行时间、递归跟快速排序。明天还是得先把计算机网络的东西看了，C++类剩余的东西争取后天搞定。