C++程序设计

C++ Programming

实验报告 5-6

Experiment Report

专业(Major)： 计算机科学与技术 2502 班

学号(ID)：202512898

姓名(Name)：田佩宁

提交日期(Date)：2025 年 12 月 21 日

成绩(Score)：

东北大学悉尼智能科技学院

Sydney Smart Technology College, Northeastern University

实验编号（Experiment No.）：5

实验名称（Experiment Name）：运算符重载的实现

实验目的（Experiment Purpose）：

（1）理解运算符重载的作用；

（2）掌握运算符重载方法。

实验内容（Experiment Contents）：

编写一个程序，定义一个时间类 Time，包含三个属性： hour, minute 和 second

要求通过运算符重载实现如下功能:

时间输入输出(>>、<<)；

时间增加减少若干(+=、-=)，例：Time& operator+=(const Time&);Time& operator-=(const Time&)；

时间前、后自增加/减少 1 秒(++、--)，前自增例：Time& operator++(); 后自增例：Time operator++(int)；

实验结果（Experiment Results）：

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

class Time {
private:
    int hour, minute, second;
    void normalize() {                  // 统一
        int carry = second / 60;
        if (second < 0) {
            carry = (second-59)/60;
            second += (-carry)*60;
        }
        minute += carry, second %= 60;

        carry = minute/60;
        if (minute < 0) {
            carry = (minute-59)/60;
            minute += (-carry)*60;
        }
        hour += carry, minute %= 60;

        hour %= 24;
        if (hour < 0) { hour += 24; }
    }
public:
    Time() : hour(0), minute(0), second(0) {}
    Time(int hour, int minute, int second)
    : hour(hour), minute(minute), second(second) {
        normalize();
    }

    /* 输入输出 */
    friend istream& operator>>(istream& in, Time& t) {
        return in >> t.hour >> t.minute >> t.second;
    }
    friend ostream& operator<<(ostream& out, const Time& t)
        return out
        <<setw(n: 2)<<setfill(c: '0')<<t.hour<<':'
        <<setw(n: 2)<<setfill(c: '0')<<t.minute<<':'
        <<setw(n: 2)<<setfill(c: '0')<<t.second<<'\n';
    }

    /* 复合赋值 +=  -= */
    Time& operator+=(const Time& rhs) {
        hour += rhs.hour;
        minute += rhs.minute;
        second += rhs.second;
        normalize(); return *this;
    }
    Time& operator-=(const Time& rhs) {
        hour -= rhs.hour;
        minute -= rhs.minute;
        second -= rhs.second;
        normalize(); return *this;
    }

    /* 前自增/自减 */
    Time& operator++() {
        second++; normalize();
        return *this;
    }
    Time& operator--() {
        second--; normalize();
        return *this;
    }

    /* 后自增/自减 */
    Time operator++(int) {
        Time tmp=*this; ++(*this);
        return tmp;
    }
    Time operator--(int) {
        Time tmp=*this; --(*this);
        return tmp;
    }
};

int main() {
    Time t1, t2;
    cin >> t1 >> t2;

    cout << (t1 += t2++) << '\n';
    cout << (t1 -= t2)   << '\n';
    cout << (++t2)       << '\n';
    cout << (t2 += t1--) << '\n';
    cout << (--t1)       << '\n';
    cout << (t2 -= t1)   << '\n';


    cout<<"田佩宁 202512898\n";
}
```

```
21 10 35
10 15 25
07:26:00

21:10:34

10:15:27

07:26:01

21:10:32

10:15:29

田佩宁 202512898
```

心得体会（Experience）：

运算符重载让代码更直观，像自然语言。实验中发现语法细节多，需严谨，体会封装与可读性的平衡。

实验编号（Experiment No.）：6

实验名称（Experiment Name）：函数模板与类模板的使用

实验目的（Experiment Purpose）：

（1）掌握函数模板的形式和使用方法；

（2）掌握类模板的形式和使用方法。

实验内容（Experiment Contents）：

模板是实现代码重用机制的一种工具，如下示例代码所示：

```
#include <iostream>
using namespace std;
template <typename T>
T max1(T a,T b)
{
    return (a>b) ? a:b;
}

int main()
{
    int a=22,b=55;
    float a1=34.5,b1=32.8;
    cout<< max1(a,b)<<endl;
    cout<< max1(a1,b1)<<endl;
    return 0;
}
```

第 1 题：利用函数模板实现对整型数组（int_arr[]={5,55,2,22,4,77}）和字符数组（char_arr[]=”nsajlkds”）的排序。（必做题）

类模板允许用户为类定义一种模式，使得类中的某些数据成员，某些成员函数的参数或返回值，能取任意数据类型。如下为一个栈类模板的使用：

```
#include <iostream>
using namespace std;
const int size =10;
template<class Type>
class stack{
  public:
     void init(){ tos=0;}
     void push(Type ch); //参数取 Type 类型
     Type pop();              //返回类型取 Type 类型
  private:
     Type stck[size];       //数组的类型为类型参数 Type
     int tos;
}

template <class Type>  //每个成员函数都要加上模板声明
void stack<Type>::push(Type ob)  //类名为 stack<Type>
{
    if (tos == size)
    {cout<<"stack is full"; return}
```

```
    stck[tos]=ob;
    tos++;
}

template <class Type>
Type stack <Type>::pop(){
   if (tos == 0)
   { cout<<"stack is empty"; return 0;}
   tos--;
   return stck[tos];
}

int main(){
   //定义字符堆栈
   stack<char> s1,s2;  //创建两个模板参数为 char 型的对象
   int i1;
   s1.init(); s2.init();
   s1.push('a');  s2.push('x');
   s1.push('b');  s2.push('y');
   s1.push('c');  s2.push('z');
   for(i1=0;i1<3;i1++) cout<<"pop s1: "<<s1.pop()<<endl;
   for(i1=0;i1<3;i1++) cout<<"pop s2: "<<s2.pop()<<endl;
   //定义整型堆栈
   stack<int> is1,is2;  //创建两个模板参数为 int 型的对象
   int i2;
   is1.init(); is2.init();
   is1.push(1);  is2.push(2);
   is1.push(3);  is2.push(4);
   is1.push(5);  is2.push(6);
   for(i2=0;i2<3;i2++) cout<<"pop is1: "<<is1.pop()<<endl;
   for(i2=0;i2<3;i2++) cout<<"pop is2: "<<is2.pop()<<endl;
   return 0;
}
```

2、编写一个类模板，完成对不同数据类型的数组的排序（小到大）操作。

```
int array[5]={3,6,2,1,4};
double array[5]={3.1,1.2,4.5,1.1,0.2};
char array[5]={ ‘s’ , ‘d’ , ‘a’ , ‘y’ , ‘t’ };
```

实验结果（Experiment Results）：

题目一

```cpp
#include <iostream>
using namespace std;

template<typename T>
T* sort(T* a, int left, int right) {
    int i = left, j = right;
    T flag = a[(left + right) / 2], temp;
    do {
        while (a[i] < flag) i++;
        while (a[j] > flag) j--;
        if (i <= j) {
            temp = a[i]; a[i] = a[j]; a[j] = temp;
            i++; j--;
        }
    } while (i <= j);
    if (left < j) sort(a, left, right: j);
    if (i < right) sort(a, left: i, right);
    return a;
}

template<typename T>
void print(T* a, int len) {
    cout << "before sort: ";
    for (int i = 0; i < len; i++) {
        cout << a[i] << (i == len - 1 ? '\n' : ' ');
    }
    cout << "after sort: ";
    sort(a, left: 0, right: len - 1);
    for (int i = 0; i < len; i++) {
        cout << a[i] << (i == len - 1 ? '\n' : ' ');
    }
}

int main() {
    int num[] = {[0]=5, [1]=55, [2]=2, [3]=22, [4]=4, [5]=77
    char str[] = {[0]='n', [1]='s', [2]='a', [3]='j', [4]='l

    print(a: num, len: 6);
    cout << '\n';
    print(a: str, len: 8);

    cout<<"田佩宁 202512898\n";
}
```

```
before sort: 5 55 2 22 4 77
after sort: 2 4 5 22 55 77

before sort: n s a j l k d s
after sort: a d j k l n s s
田佩宁 202512898
```

题目二

```cpp
#include <iostream>
using namespace std;

const int DefaultLen = 5;

template<typename T>
class DataProcess{
    T* a;
    int len;
public:
    DataProcess<T>(T* array, int l=DefaultLen)
    :a(array), len(l){}
    void sort(T*&a, int left, int right) {
        int i = left, j = right;
        T flag = a[(left + right) / 2], temp;
        do {
            while (a[i] < flag) i++;
            while (a[j] > flag) j--;
            if (i <= j) {
                temp = a[i]; a[i] = a[j]; a[j] = temp;
                i++; j--;
            }
        } while (i <= j);
        if (left < j) sort(&: a, left, right: j);
        if (i < right) sort(&: a,left: i, right);
    }
    void print() {
        cout << "before sort: ";
        for (int i = 0; i < len; i++) {
            cout << a[i] << (i == len - 1 ? '\n' : ' ');
        }
        cout << "after sort: ";
        sort(&: a, left: 0, right: len - 1);
        for (int i = 0; i < len; i++) {
            cout << a[i] << (i == len - 1 ? '\n' : ' ');
        }
    }
};
int main() {
    int num1[] = {[0]=3, [1]=6, [2]=2, [3]=1, [4]=4};
    DataProcess<int>_num1(array: num1);
    double num2[] = {[0]=3.1, [1]=1.2, [2]=4.5, [3]=1.1, [4]
    DataProcess<double>_num2(array: num2);
    char str[] = {[0]='s', [1]='d', [2]='a', [3]='y', [4]='t
    DataProcess<char>_str(array: str);

    _num1.print();  cout<<'\n';
    _num2.print();  cout<<'\n';
    _str.print();   cout<<'\n';

    cout<<"田佩宁 202512898\n";
}
```

```
before sort: 3 6 2 1 4
after sort: 1 2 3 4 6

before sort: 3.1 1.2 4.5 1.1 0.2
after sort: 0.2 1.1 1.2 3.1 4.5

before sort: s d a y t
after sort: a d s t y

田佩宁 202512898
```

心得体会（Experience）：

模板让代码复用更高效，类模板与函数模板结合，泛型思维提升，编译期检查更严谨。