第四次实验报告 技术预览版

第一部分：实验题目简注

1. 第9章课后题7

2. 第9章课后题9

3. 第9章课后题12

4. 设计两个函数模板sort() 和 print()

5. 设计一个数组类模板Array

6. 让Stack类更加完美

7. 关于<<操作符不能定义为成员函数的问题

8. 关于有些运算符只能定义为成员函数的问题

第二部分：实验细节

1.第9章课后题7 常量的问题

注：这个已经在上次实验报告的“关于const”的总结中已经有过详细的描述。

*=====引用上次实验报告的内容=====*

*4.关于const的总结*

*const常常出现在变量限定，指针限定，函数限定的地方。const具有“常的含义”，其作用可以通俗地理解为保持…不变的特性。*

*① 变量定义时(非指针)加const，实际上就是定义一个常对象（一般来说，常基类型数据也属于常对象），其所存储的数据不可更改。在简单的int等类型加const，其数据不可更改，在对象前加const，则其数据（即成员）不可更改。因为对象的数据不可更改，所以定义对象时必须对其初始化，常对象不可调用非const函数。*

例子 const int a = 12; const student a(1,张三);

*② 指针定义时，可以在两处地方加const，在类型前加const表示其指向的数据不可更改（可以用变量的地址赋值，但不能通过指针修改其指向的值），而在指针前加const表示其指向不可更改（这与引用有相似指出）。*

*例如:*

int a=1,b=2;

const int \*pa = &a;

pa = &b;

\*pa = 4;

int a=1,b=2;

const int \* const pa = &a;

pa = &b;

*③ 当函数用const修饰时，其声明该函数不会修改数据成员。当一个函数没有修改数据成员但没有标识为const时，const对象也无法调用其值。*

*④ 在拷贝构造函数调用时，其参数定义为只读的引用，所以在拷贝构造函数调用时，不能访问其非const数据成员。同时，为了告诉用户这个方法不会随意修改数据，通常也会将这些方法标识为const.*

*==========*

这些是新添加的内容

① const类型数据的初始化