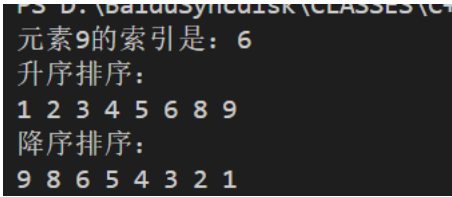
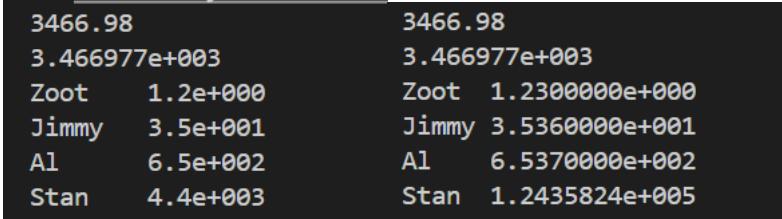
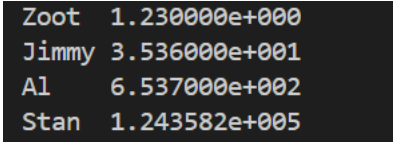


计算机学院 高级语言程序设计 课程实验报告

实验题目：STL, 流文件综合练习		学号：202200400053
日期：2024-06-06	班级：2202	姓名：王宇涵
Email： 1941497679@qq.com		
实验目的： 1. 练习 STL 编程 2. 练习流类库中常用的类及成员函数的用法， 3. 练习标准输入/输出流及格式控制。 4. 练习文件流的操作（文本文件、二进制文件）		
实验软件和硬件环境： 软件环境：VSCODE + DEV-C++ 硬件环境：Legion Y7000P		
实验步骤与内容： 1. 分析运行第 10 章实验任务（2），lab10_2.cpp, 熟悉 STL 的使用。  分析: int a[8] = {3, 2, 5, 6, 4, 1, 9, 8}; 通过 find 函数进行查找元素 9, 返回索引 6 通过 sort 函数进行排序, 默认升序排序, 后指定降序排序 greater<int>(). 2. 格式控制 (1) 实践第 11 章 PPT, 例 11-4, 格式控制, 精度 setprecision()用法 如果将 value 中的 4358.24 改为 124358.24; 再将 setprecision(1)改为 setprecision(7),看输出结果如何?  分析：使用了控制精度 setprecision(), 也使用了 scientific, setw, resetiosflags(ios_base::left)等控制科学计数法, 宽度, 左对齐等. 修改后输出结果如右图, 精度到了小数点的后 7 位. 如果去掉 setprecision(); 输出如何?  去掉 setprecision()后, 输出如上, 此时没有控制精度		

(2) 完成学生用书, 第 11 章实验 11, 实验任务 (1), 运行 lab11_1.cpp, 用记事本打开该程序输出的文本文件, 分析其中输出信息。

```
int i = 53;
float f = 4700113.141593;
T.setf(ios::unitbuf);
T.setf(ios::showbase);
T.setf(ios::uppercase);
T.setf(ios::showpos);
T << i << endl;
+53
T.setf(ios::hex, ios::basefield);
T << i << endl;
0X35
T.unsetf(ios::uppercase);
T.setf(ios::oct, ios::basefield);
T << i << endl;
065
T.unsetf(ios::showbase);
T.setf(ios::dec, ios::basefield);
T.setf(ios::left, ios::adjustfield);
T.fill('0');
T << "fill char: " << T.fill() << endl;
fill char: 0
T.width(8);
+5300000
T.setf(ios::right, ios::adjustfield);
T.width(8);
00000+53
T.setf(ios::internal, ios::adjustfield);
T.width(8);
+0000053
T << i << endl;
+53

4.700113e+006
T.setf(ios::fixed, ios::floatfield);
T << f << endl;
4700113.000000
T << f << endl;
4700113.000000
T.precision(16);
T << "prec = " << T.precision() << endl;
prec = 16
T << endl << f << endl;

4700113.0000000000000000
T.setf(ios::scientific, ios::floatfield);
T << endl << f << endl;

4.7001130000000000e+006
T.setf(ios::fixed, ios::floatfield);
T << f << endl;
4700113.0000000000000000
T << f << endl;
4700113.0000000000000000
T.width(8);
Is there any more?
T.width(36);
0000000000000000Is there any more?
T.setf(ios::left, ios::adjustfield);
T.width(36);
Is there any more?0000000000000000
T.unsetf(ios::showpoint);
T.unsetf(ios::unitbuf);
```

分析：功能是先将本身的代码字符信息传入文件，若代码本身有信息传入文件的操作，则显示传入的信息。

举例：先定义了 $i = 53$ ，后将 i 输出到文件中，则在 $T \ll i \ll endl$ 信息下，出现 +53 的信息。

后续进行了很多操作，包含了进制转化，基数设置，对齐方式，输出宽度等等。

3. 实践第 11 章 PPT, 例 11-5, 二进制文件输出。如果用记事本程序打开它的输出文件，能看懂其中的信息吗？再换用其他可十六进制显示文件内容的软件（如 EditPlus 软件，Hx: Hex viewer 模式）查看其内容。



\

看不懂信息，换做十六进制显示文件内容的软件

使用 windows cmd 命令 certutil -encodehex date.txt hello.txt 将 date.txt 文件转化为 hello.txt 文件

如图为文件内容

```
0000 06 00 00 00 0a 00 00 00 5c 00 00 00 ..... \...
```

4. 实践第 11 章 PPT, 例 11-6, 字符串输出流 (ostringstream)。

5

1.2

PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp12\code\output>

原理：通过 ostringstream 将初始化输入的数字 5 和 1.2 转化为字符串，进行打印

5. 文本文件

(1) 实践第 11 章 PPT, 例 11-8 (getline 函数读文本文件)。试用不同的输入方式（含空格），解释

结果。

试用 `get` 函数完成同样的功能。

```
Type a line terminated by 't'
ddfwf t
ddfwf
```

```
Type a line terminated by 't'
12e1oet
12e1oe
```

```
Type a line terminated by 't'
fdwfw b dt
fdwfw b d
```

解释结果：`getline` 函数会读取字符(包含空格)，一直读到 `t` 为止，再将 `t` 删除。

用 `get` 函数完成功能：

```
string line;
// cout << "Type a line terminated by 't' " << endl;
// getline(cin, line, 't');
char ch;
cout << "Type a line terminated by 't' " << endl;
while (cin.get(ch) && ch != 't') {
    line += ch;
}

cout << line << endl;
return 0;
```

测试

```
Type a line terminated by 't'
faffw bt
faffw b
```

(2) 完成学生用书，习题 11-5，编程实现上面第 5 题中的文本文件内容添加、读入、输出。

实现代码

```
int main()
{
    ofstream file1("test.txt");
    file1 << "已成功写入文件! ";
    file1.close();
    file1.open("test.txt", ios::app);
    file1 << "已成功添加字符! ";
    file1.close();
    char ch;
    ifstream file2("test.txt");
    while (file2.get(ch))
        cout << ch;
    file2.close();
    return 0;
}
```

测试结果

```
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp12\code\output> & .\ex11-5.exe
已成功写入文件! 已成功添加字符!
```

6. 二进制文件

参考第 11 章 PPT，例 11-9（write、read 函数 写、读二进制文件），将对象存储磁盘文件。
完成学生用书，习题 11-6，练习文本文件、二进制文件的输入、输出。

```
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp12\code\output> .\
Please enter the file name: dog.txt
dog2 weight: 2
dog2 days: 2
dog2 weight: 5
dog2 days: 10
PS D:\BaiduSyncdisk\CLASSES\C++\exp\exp12\code\output> █
```

分析：先创建了 dog1(5, 10)用二进制的形式写入文件，再创建了 dog2(2, 2)进行信息输出，因此输出 (2, 2)；再用二进制的形式读入文件，将 dog1 信息输入 dog2 中，因此输出(5, 10)

7. 练习习题 11-9，练习文本文件读入，（添加行号、格式控制右对齐）输出。

Input.txt

```
hi
hello
```

Output.txt

```
1. hi
2. hello
|
```

分析：先将 input.txt 的内容输入字符流，再设置输出流的对齐方式为右对齐，设置输出宽度为 1（确保行号的宽度），并输出行号和行内容且换行，逐行读取字符流中的内容到 output.txt 中。

8. 综合实例：课本 11.5 节，银行账户例子，例 11-13。

```
2008-11-1      Total: 0      command> a s 001 0.015
2008-11-1      #001 created
2008-11-1      Total: 0      command> a c 002 5000 0.02 50
2008-11-1      #002 created
2008-11-1      Total: 0      command> d 0 1000 存入1000元到储蓄账户001
2008-11-1      #001      1000      1000      存入1000元到储蓄账户001
2008-11-1      Total: 1000   command> d 1 2000 存入2000元到信用账户002
2008-11-1      #002      2000      2000      存入2000元到信用账户002
2008-11-1      Total: 3000   command> w 0 200 从储蓄账户001取出200元
2008-11-1      #001      -200      800      从储蓄账户001取出200元
2008-11-1      Total: 2800   command> w 1 100 从信用账户002取出100元
2008-11-1      #002      -100      1900     从信用账户002取出100元
2008-11-1      Total: 2700   command> s
[0] 001 Balance: 800
[1] 002 Balance: 1900   Available credit:5000
2008-11-1      Total: 2700   command> c 15
2008-11-15      Total: 2700   command> n
2008-12-1      Total: 2700   command> q 2008-11-01 2008-12-01
2008-11-1      #001      1000      1000      存入1000元到储蓄账户001
2008-11-1      #002      2000      2000      存入2000元到信用账户002
2008-11-1      #001      -200      800      从储蓄账户001取出200元
2008-11-1      #002      -100      1900     从信用账户002取出100元
2008-12-1      Total: 2700   command> e
```

测试样例：

增加一个储蓄账户，ID 为 001，利率为 0.015。

增加一个信用账户，ID 为 002，信用额度为 5000，利率为 0.02，年费为 50。

向储蓄账户 001 存入 1000 元。

向信用账户 002 存入 2000 元。

从储蓄账户 001 取出 200 元。

从信用账户 002 取出 100 元。

显示所有账户信息。

将当前日期改变为本月的 15 日。

进入下个月，并结算所有账户。

查询从 2008 年 11 月 1 日到 2008 年 12 月 1 日的账目信息。

退出程序。

结论分析与体会：

通过这次实验，我深入学习并应用了 C++ 的 STL 和格式控制，熟悉了 `find` 和 `sort` 函数的使用，掌握了如何查找元素并进行排序，包括升序和降序排序。

通过实践第 11 章的例子，我学会了使用 `setprecision` 来控制输出精度，以及如何使用 `scientific`、`setw`、`resetiosflags(ios_base::left)` 等方法进行格式控制。这些知识帮助我理解了如何在程序中精确控制输出格式和数值显示。

此外，通过对文本文件和二进制文件的读写操作，我进一步了解了文件操作的细节，尤其是如何使用 `write` 和 `read` 函数进行二进制文件的处理。

在银行账户管理系统的综合实例中，我通过测试样例验证了账户的增加、存取款、账户信息显示、日期变更、结算和账目查询等功能，熟悉了如何设计和实现一个完整的应用程序。

这次实验不仅巩固了我对 C++ 语言的基础知识，还提升了我对实际问题的解决能力，增强了我的编程技巧和调试能力。

就实验过程中遇到的问题及解决处理方法，自拟 1—3 道问答题：

1. STL 编程的优势？

答：STL 编程的优势在于它提供了一组功能强大、灵活且高效的模板类和函数，极大地简化了数据结构和算法的实现。STL 还利用模板机制实现了泛型编程，使得代码的复用性更高，类型安全性更强。在开发过程中，STL 的使用能够减少代码量，降低错误率，提高程序的可靠性和执行效率，显著提升编程的生产力。

2. 使用二进制进行文件输入输出有什么好处？

答：二进制文件的读写操作通常比文本文件更快。因为二进制文件直接以字节形式存储数据，而不需要进行字符编码转换或格式解析，文件的体积也更小，因此减少了 CPU 的开销。此外，二进制文件可以精确地存储浮点数、整数等数据类型，而不损失任何精度。在文本文件中，浮点数转换为字符串后再转换回浮点数，可能会导致精度损失。

3. Getline()和 get()函数的区别？

答：`getline()` 函数用于从输入流中读取一行数据，并将其存储为字符串。相比之下，`get()` 函数用于从输入流中读取一个字符。

总的来说，`getline()` 适合读取整行文本数据，而 `get()` 则更适合逐字符地读取输入流。