计算机学院 高级程序语言设计 课程实验报告

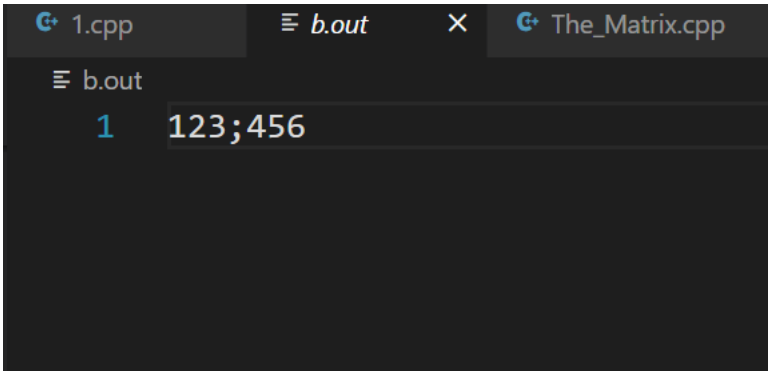
实验题目：流类库与输入输出流		学号：202300130150
日期：2024.5.16	班级： 23.4	姓名： 王成意

实验目的：

1. 练习流类库中常用的类及成员函数的用法。
2. 练习标准输入/输出及格式控制
3. 练习文件的操作（文本文件、二进制文件）

实验步骤与内容：

1. 实践第 11 章 PPT，P7，标准输出换向到文件。
   运行结果：终端无输出，但是文件夹多出了 b.out 文件，打开发现：



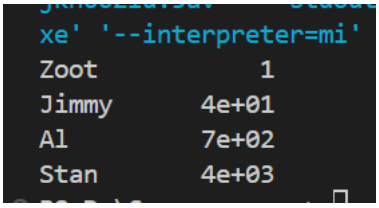
，符合预期。

2. 2. 实践第 11 章 PPT，例 11-4， 格式控制，精度 setprecision()用法
实践代码：

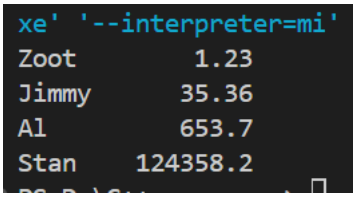
```
// 11_4.cpp

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;



int main()

{
```

```
    double values[] = {1.23, 35.36, 653.7, 4358.24};
    string names[] = {"Zoot", "Jimmy", "Al", "Stan"};
    for (int i = 0; i < 4; i++)
        cout << setiosflags(ios_base::left)
             << setw(6) << names[i]
             << resetiosflags(ios_base::left) // 清除左对齐设置
             << setw(10) << setprecision(1) << values[i] << endl;
    return 0;
}
```

实践结果：



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？



，可见最后舍弃了一位数。

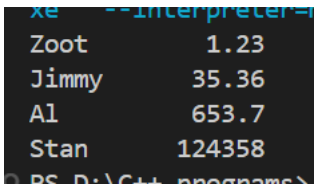
如果去掉 setprecision()；输出如何？

代码 1：

```
// 11_4.cpp
#include <iostream>
#include <iomanip>
```

```cpp
#include <string>

using namespace std;


int main()

{

    double values[] = {1.23, 35.36, 653.7, 124358.24};

    string names[] = {"Zoot", "Jimmy", "Al", "Stan"};

    for (int i = 0; i < 4; i++)

        cout << setiosflags(ios_base::left)

             << setw(6) << names[i]

             << resetiosflags(ios_base::left) // 清除左对齐设置

             << setw(10) << values[i] << endl;

    return 0;

}
```

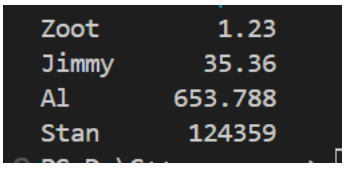
结果：



代码 2：

```cpp
// 11_4.cpp

#include <iostream>

#include <iomanip>

#include <string>
```

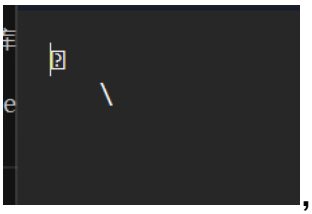
```cpp
using namespace std;


int main()
{
    double values[] = {1.23, 35.36, 653.7879,
124358.84};
    string names[] = {"Zoot", "Jimmy", "Al", "Stan"};
    for (int i = 0; i < 4; i++)
        cout << setiosflags(ios_base::left)
             << setw(6) << names[i]
             << resetiosflags(ios_base::left) // 清除左
对齐设置
             << setw(10) << values[i] << endl;
    return 0;
}
```

结果： 

，由此可见不加 setprecision()，输出 6 位有效数字并且默认进行四舍五入。

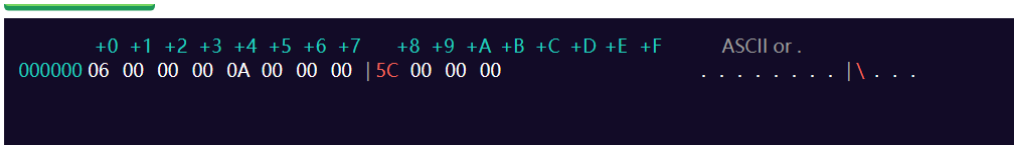
3. 实践第 11 章 PPT，例 11-5，二进制文件输出。如果用记事本程序打开它的输出文件，能看懂其中的信息吗？再换用其他可十六进制显示文件内容的软件（如 EditPlus 软件，Hx：Hex viewer 模式）查看其内容。

记事本打开：



，显然不可读。

用十六进制显示如下：



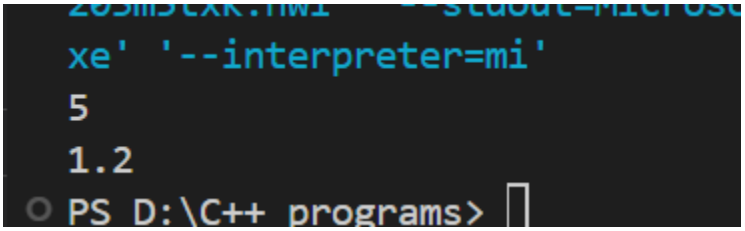
简单分析可知，分别对应 6，10，92.
（5*16+12（c）=92）

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

```
// 11_6.cpp

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

template <class T>

inline string toString(const T &v)

{

    ostringstream os; // 创建字符串输出流

    os << v;          // 将变量 v 的值写入字符串流

    return os.str();  // 返回输出流生成的字符串

}

int main()
```

```
{
    string str1 = toString(5);
    cout << str1 << endl;
    string str2 = toString(1.2);
    cout << str2 << endl;
    return 0;
}
```

测试结果：



，符合预期。

5. 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。

代码：

```
// 11_8.cpp
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string line;
    cout << "Type a line terminated by 't' " << endl;
    getline(cin, line, 't');
```

```
    cout << line << endl;

    return 0;

}
```

结果：

```
 xe' '--interpreter=mi'
 Type a line terminated by 't'
 dadasdsada00001111tttt
 dadasdsada00001111
 PS D:\C++ programs>
```

，可见输入到 t 之前的字符被放在了 line 里输出。

试用 get 函数完成同样的功能。

代码：

```
// 11_8.cpp

#include <iostream>

#include <string>

using namespace std;

int main()

{

    string line;

    cout << "Type a line terminated by 't' " << endl;

    char xxx;

    while((xxx=cin.get())!='t')

    {

        line += xxx;

    }
```

```
    cout << line;

    return 0;

}
```

结果：

```
xe' '--interpreter=mi'
Type a line terminated by 't'
11223344qqaazzxxsswwetttq;sod
11223344qqaazzxxsswwe
PS D:\C++ programs>
```

6. 实践第 11 章 PPT，例 11-9（write、read 函数，块方式写、读二进制文件），将对象存储磁盘文件。

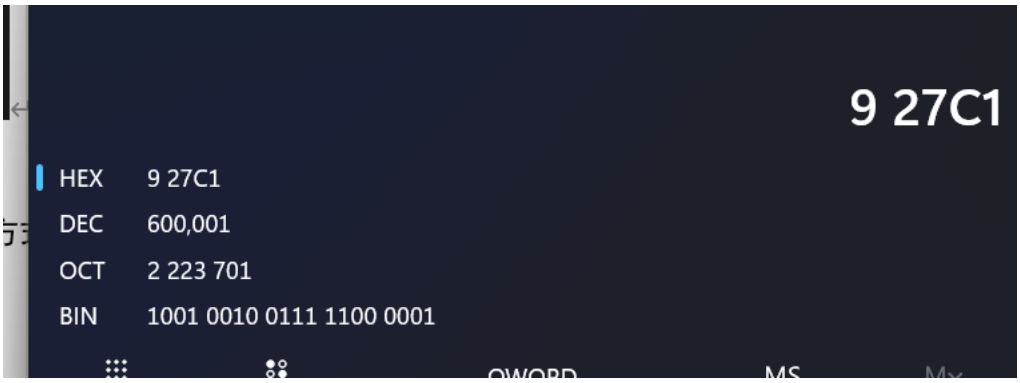
运行结果：

```
xe' '--interpreter=m
600001 8000
PS D:\C++ programs>
```

查看 payroll 文件：

```
        +0 +1 +2 +3 +4 +5 +6 +7   +8 +9 +A +B +C +D +E +F      ASCII or .
000000 C1 27 09 00 00 00 00 00 |00 00 00 00 00 40 BF 40      . ' . . . . . |. . . . . @. @
```

其中，9 27 c1 代表 600001：



后面对应的 8000 应该是 1F40，但是没有成功解析；

换成 16000：

```
   +0 +1 +2 +3 +4 +5 +6 +7   +8 +9 +A +B +C +D +E +F      ASC
0 C1 27 09 00 00 00 00 00 |00 00 00 00 00 40 CF 40      . ' .
```

由 BF 变成了 CF，可见其拆位方式有不同，但是在某些逻辑上是一样的。

8. 定义一个 Dog 类，包含体重和年龄两个成员变量及相应的成员函数，声明一个实例 dog1，体重为 5，年龄为 10，使用 I/O 流把 dog1 的状态写入磁盘文件，再声明另一个实例 dog2，通过读文件把 dog1 的状态赋给 dog2。分别使用

文本方式和二进制方式操作文件，看看结果有何不同；再看看磁盘文件的ASCII 码有何不同。
代码：

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

class Dog
{
private:
    double weight;
    int age;

public:
    Dog();
    Dog(double weight, int age);
    void setWeight(double weight);
    void setAge(int age);
    double getWeight() const;
    int getAge() const;
    void writeToFileText(const std::string &filename)
const;
```

```
    void readFromFileText(const std::string
&filename);
    void writeToFileBinary(const std::string
&filename) const;
    void readFromFileBinary(const std::string
&filename);
};

double Dog::getWeight() const
{
    return weight;
}

int Dog::getAge() const
{
    return age;
}

Dog::Dog()
{
    this->weight = 0.0;
    this->age = 0;
```

```
}

Dog::Dog(double weight, int age)
{
    this->weight = weight;
    this->age = age;
}

void Dog::setWeight(double weight)
{
    this->weight = weight;
}

void Dog::setAge(int age)
{
    this->age = age;
}

void Dog::writeToFileText(const string &filename)
const
{
    std::ofstream outputFile(filename);
```

```
    if (outputFile.is_open())
    {
        outputFile << weight << "\n";
        outputFile << age << "\n";
        outputFile.close();
        cout << "Dog writeToFileText success " <<
filename << std::endl;
    }
    else
    {
        std::cerr << "WA on Dog::writeToFileText " <<
std::endl;
    }
}

void Dog::readFromFileText(const string &filename)
{
    std::ifstream inputFile(filename);
    if (inputFile.is_open())
    {
        inputFile >> weight;
        inputFile >> age;
```

```
        inputFile.close();
        cout << "Dog::readFromFileText success " <<
filename << std::endl;
    }
    else
    {
        std::cerr << "WA on Dog::readFromFileText " <<
std::endl;
    }
}

void Dog::writeToFileBinary(const string &filename)
const
{
    std::ofstream outputFile(filename,
std::ios::binary);
    if (outputFile.is_open())
    {
        outputFile.write(reinterpret_cast<const char
*>(&weight), sizeof(weight));
        outputFile.write(reinterpret_cast<const char
*>(&age), sizeof(age));
```

```
        outputFile.close();
        cout << "Dog::writeToFileBinary sueecss " <<
filename << std::endl;
    }
    else
    {
        std::cerr << "WA on Dog::writeToFileBinary "
<< std::endl;
    }
}


void Dog::readFromFileBinary(const std::string
&filename)
{
    std::ifstream inputFile(filename,
std::ios::binary);
    if (inputFile.is_open())
    {
        inputFile.read(reinterpret_cast<char
*>(&weight), sizeof(weight));
        inputFile.read(reinterpret_cast<char *>(&age),
sizeof(age));
```

```
        inputFile.close();
        cout << "Dog::readFromFileBinary success" <<
filename << std::endl;
    }
    Else


    {
        std::cerr << "WA on Dog::readFromFileBinary"
<< std::endl;
    }
}
int main()
{
    Dog dog1(6.6,6);
    dog1.writeToFileText("dog1.txt");
    dog1.writeToFileBinary("dog1.bin");
    Dog dog2;
    dog2.readFromFileText("dog1.txt");
    cout << "dog from txt" << std::endl;
    cout << dog2.getAge() << " " << dog2.getWeight()
<< endl;
    dog2.readFromFileBinary("dog1.bin");
```

```
    cout << "dog from bin" << std::endl;

     cout << dog2.getAge() << " " << dog2.getWeight()
<< endl;

}
```

结果：

```
Dog writeToFileText success dog1.txt
Dog::writeToFileBinary sueecss dog1.bin
Dog::readFromFileText success dog1.txt
dog from txt
6 6.6
Dog::readFromFileBinary successdog1.bin
dog from bin
6 6.6
```

附某一次的失败截图：

```
Dog writeToFileText successdog1.txt
Dog::writeToFileBinary sueecssdog1.bin
WA on Dog::readFromFileText
dog from txt
0 0
WA on Dog::readFromFileBinary
dog from bin
0 0
PS D:\C++ programs> ^C
PS D:\C++ programs>
```

再看看磁盘文件的 ASCII 码有何不同：

Dog.txt:

```
ASCII or .
6 . 6 . . 6 . . |
```

Dog.bin:

```
ASCII or .
f f f f f . @|. . . .
```

结论分析与体会：

非常好的实验，让我进一步认识并且练习与掌握了流类库中常用的类及成员函数的用法，标准输入/输出及格式控制和文件的操作。