

第三次作业实验报告

源代码见压缩文件包

第一题：测试各种常见类型的字节数并设计一种方法测试float类型所能保存的小数位数

思路

直接对各种常见类型使用sizeof()运算符即可

对测试float类型，可以用一无限循环小数不断乘以10后与相应的数比较知道有几位精确保存（float类型乘以10不精确但调用Visual Studio 2017的调试模式发现不影响1/3存储的精度），而又通过调用调试模式发现，当把float类型的1/3乘以足够多次10使得VS默认保存的位数全部在小数点左侧后，再次乘以10并进行类型强制转换为int时会得到负数，可由此得出VS默认保存的小数位数。

运行结果截图



```
Microsoft Visual Studio 调试控制台
the size of short:2
the size of int:4
the size of long:4
the size of unsigned short:2
the size of unsigned int:4
the size of unsigned long:4
the size of char:1
the size of bool:1
the size of double:8
the size of float:4
在VS中float类型保留的精确小数位数为: 7
在VS中float类型默认总共保存的小数位数为: 9
D:\Files\课程\程设\作业\homework\Debug\homework.exe (进程 42312) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      cout << "the size of short:" << sizeof(short) << endl << "the size of int:" <<
        sizeof(int) << endl;
```

```

6      cout << "the size of long:" << sizeof(long) << endl << "the size of unsigned
short:" << sizeof(unsigned short) << endl;
7      cout << "the size of unsigned int:" << sizeof(unsigned int) << endl << "the
size of unsigned long:" << sizeof(unsigned long) << endl;
8      cout << "the size of char:" << sizeof(char) << endl << "the size of bool:" <<
sizeof(bool) << endl;
9      cout << "the size of double:" << sizeof(double) << endl << "the size of float:"
<< sizeof(float) << endl;
10
11     float fTest = 0;
12     int iNum = 0, iTest = 3;
13     fTest = 1.0 / 3.0;
14     for (int i = 1;; i++) {
15         fTest *= 10;
16         if ((int)(fTest - iTest) == 0) { iNum++; iTest = iTest * 10 + 3; }
17         else break;
18     }
19     fTest = 1.0 / 3.0;
20     int iNum2 = 0;
21     for (int i = 1;; i++) {
22         fTest *= 10;
23         int iTest2 = (int)fTest;
24         if ((int)(fTest - iTest2) == 0) iNum2++;
25         else break;
26     }
27     fTest = 1.0 / 3.0;
28     cout << "在vs中float类型保留的精确小数位数为: " << iNum << endl;
29     cout << "在vs中float类型默认总共保存的小数位数为: " << iNum2 << endl;
30
31     return 0;
32 }

```

分析总结

采用乘以10这种方式是不精确的，如果可以直接获取系统存储float类型的二进制进行分析可能会更好。

查阅资料发现float类型存储小数时可能第七位也会不精确，所以直接用cout输出时只输出6位。倘若用cout.precision()强制输出多位小数的话，后面会补充很多没有规律的数字，推测是把下一字节存储的数据显示了出来。

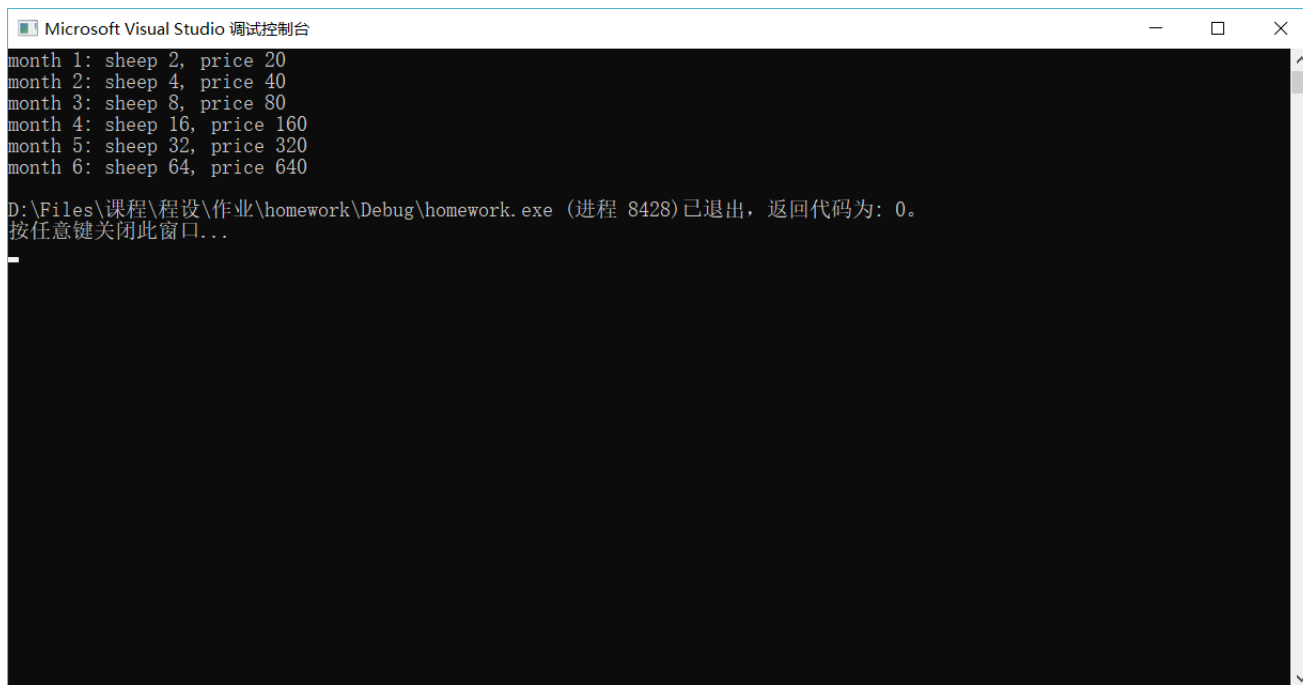
第二题：牧农缴税

思路

分析可知，一对小羊在第二个月成熟并随即生下有一对小羊，即下个月羊的数量翻倍。

所以，羊的数量满足函数 $y = 2^x$ （x表示月份），那么可卖出的钱数为 $10y$

运行结果截图



```
Microsoft Visual Studio 调试控制台
month 1: sheep 2, price 20
month 2: sheep 4, price 40
month 3: sheep 8, price 80
month 4: sheep 16, price 160
month 5: sheep 32, price 320
month 6: sheep 64, price 640
D:\Files\课程\程设\作业\homework\Debug\homework.exe (进程 8428) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

代码

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int sheep[7], price;
6      sheep[0] = 1;
7      for (int month = 1; month < 7; month++) {
8          sheep[month] = sheep[month - 1] * 2;
9          price = sheep[month] * 10;
10         cout << "month " << month << ": sheep " << sheep[month] << ", price " <<
11         price << endl;
12     }
13     return 0;
14 }
```

分析总结

与斐波那契数列所对应的兔子繁殖问题中兔子成熟后要再过一段时间才可生育不同，本题中羊的繁殖恰满足指数函数 $y = 2^x$ ，所以只需把羊的数量不断乘以2即可。

分析结果，可知牧羊人在半个月内不能缴纳足够的税款。

第三题：存储手机号（11位整型数）

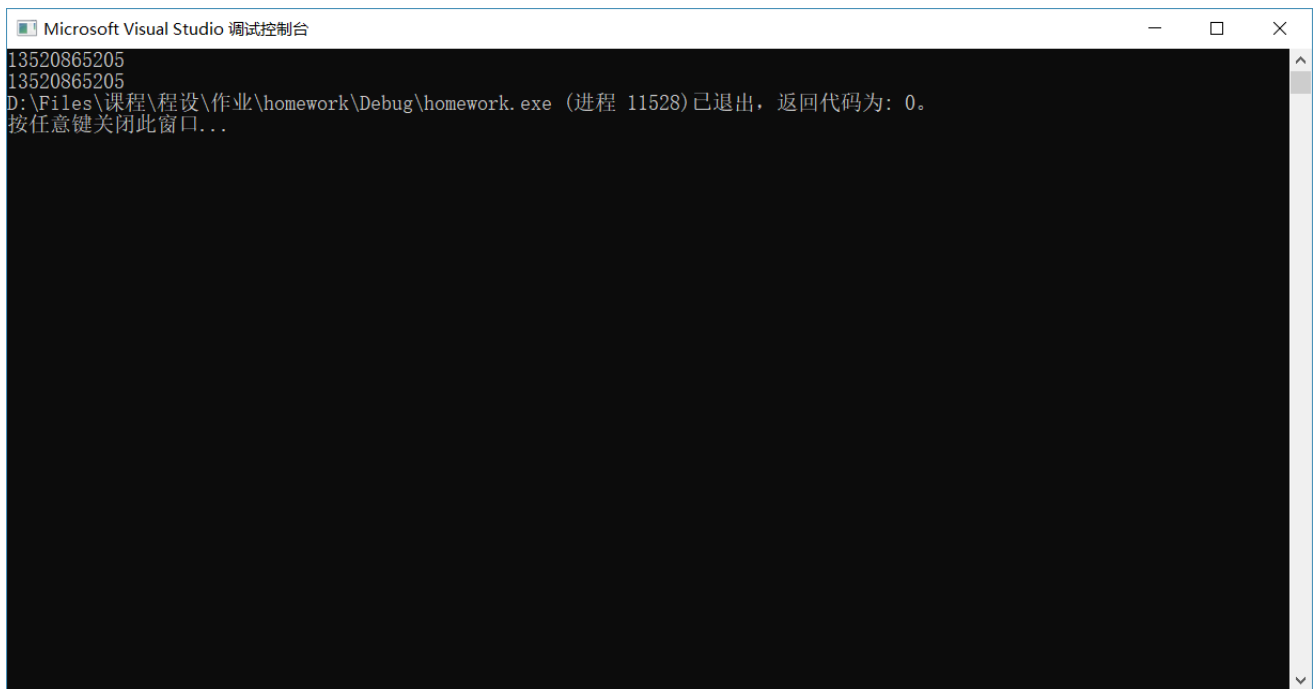
思路

存储手机号即存储11位整型数，考虑到int和long所能保存的最大数值不够，所以可以选用 long long 或者 unsigned long long 进行存储。也可以用字符数组或者字符串类型将11位整型数当作字符串进行存储。还可以采用 double，甚至 long double 和 long float 类型进行存储，不过输出时加上 fixed 控制符强制不用科学计数法，再通过cout.precision(0)使得输出中没有小数达成输出整型数的结果。

对于普通的数组来说，两元素的输入需要用空格或者回车隔开，所以无法直接使用。

unsigned long, float存储的数值不够大。

运行结果截图



```
Microsoft Visual Studio 调试控制台
13520865205
13520865205
D:\Files\课程\程设\作业\homework\Debug\homework.exe (进程 11528) 已退出，返回代码为: 0。
按任意键关闭此窗口...
```

代码

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      //为方便使用不同类型存储数据，变量名声明为 phoneNumber
7      //采用long long类型
8      //long long phoneNumber;
9
10     //采用字符数组char[]
11     //char phoneNumber[12];
12
13     //采用string类型
14     //string phoneNumber;
15
16     //采用unsigned long long类型
17     //unsigned long long phoneNumber;
18
19     //采用double类型
20     //double phoneNumber;
21     //long double phoneNumber;
22
23     //采用long float类型
24     long float phoneNumber;
25
26     cout.precision(0);
```

```
27     cin >> phoneNumber;
28     cout << fixed << phoneNumber;
29
30     return 0;
31 }
```

注：此代码只需将8、11、14、17、20行中有且仅有一行不被注释掉即可

分析总结

本题只需要尝试各种类型即可。

注意到使用 long float 和 long double 时IDE提示是非标准扩展：长浮点，但由于输入的只涉及整数部分，好像不影响结果。