

# 第九次作业实验报告

---

## 实验环境

以下所有实验都处于这一环境中

### 操作系统

Windows 10 家庭中文版 64位 版本10.0.17134.345

### 硬件

CPU Intel Core i7-8750H

RAM 8GB

### IDE

Microsoft Visual Studio Community 2017 VisualStudio.15.Release/15.8.5+28010.2036

Visual C++ 2017 00369-60000-00001-AA380

## 第一题：挑选肥羊

---

### 实验目的

设有若干只羊 (>100只)，需要从中挑选最肥的100只供使用。如何选拔这100只最肥的羊，请同学帮忙使用至少2种不同算法编程。提示：羊的重量可以采用随机函数 `rand()` 来产生。另外：使用断点调试，截获保存“最大100只羊”数组的调试窗口图，并在作业报告中提交。

初步学习排序算法

### 实验内容

#### 分析

最容易想到的就是对存储羊的权重的数组进行冒泡/选择排序，将最重的100只羊挑出来即可。

观察到 `rand()` 产生的五位数、四位数足够多，故将羊的重量初始化为0，然后再赋值。之后的挑选阶段也可以将挑出的羊赋值为0表示选过了。

#### 代码

##### 冒泡排序

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int iNumberOfSheep;
6      cout << "Please input the number of sheep: \n";
7      cin >> iNumberOfSheep;
8      int iweightOfSheep[1024] = { 0 };
9      for (int i = 0; i < iNumberOfSheep; ++i) {
10         iweightOfSheep[i] = rand();
11     }
12
13     for (int i = 0; i < 100; ++i) {
14         for (int j = iNumberOfSheep - 1; j > i; --j) {
15             if (iweightOfSheep[j] > iweightOfSheep[j - 1]) {
16                 iweightOfSheep[j - 1] += iweightOfSheep[j];
17                 iweightOfSheep[j] = iweightOfSheep[j - 1] - iweightOfSheep[j];
18                 iweightOfSheep[j - 1] -= iweightOfSheep[j];
19             }
20         }
21     }
22
23     for (int i = 0; i < 100; ++i) {
24         cout << iweightOfSheep[i] << endl;
25     }
26
27     return 0;
28 }

```

## 选择排序

```

1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int iNumberOfSheep, temp;
6      cout << "Please input the number of sheep: \n";
7      cin >> iNumberOfSheep;
8      int iweightOfSheep[1024] = { 0 }, iHeavy[100] = { 0 };
9      for (int i = 0; i < iNumberOfSheep; ++i) {
10         iweightOfSheep[i] = rand();
11     }
12
13     for (int i = 0; i < 100; ++i) {
14         for (int j = 0; j < iNumberOfSheep; ++j) {
15             if (iHeavy[i] < iweightOfSheep[j]) {
16                 iHeavy[i] = iweightOfSheep[j];
17                 temp = j;
18             }
19         }
20         iweightOfSheep[temp] = 0;
21     }

```

```

22     for (int i = 0; i < 100; ++i) {
23         cout << iHeavy[i] << endl;
24     }
25
26     return 0;
27 }

```

## 结果

局部变量			
名称	值	类型	
i	-858993460	int	
iNumberOfSheep	150	int	
iWeightOfSheep	0x00000020b90fe0e0 (41, 18467, 6334, 26500, 19169, 15724, ...	int[1024]	
[0]	41	int	
[1]	18467	int	
[2]	6334	int	
[3]	26500	int	
[4]	19169	int	
[5]	15724	int	
[6]	11478	int	
[7]	29358	int	
[8]	26962	int	
[9]	24464	int	
[10]	5705	int	
[11]	28145	int	
[12]	23281	int	
[13]	16827	int	
[14]	9961	int	
[15]	491	int	
[16]	2995	int	
[17]	11942	int	
[18]	4827	int	
[19]	5436	int	
[20]	32391	int	
[21]	14604	int	
[22]	3902	int	
[23]	153	int	
[24]	292	int	
[25]	12382	int	
[26]	17421	int	
[27]	18716	int	
[28]	19718	int	
[29]	19895	int	

自动窗口 局部变量 监视 1

局部变量

名称	值	类型
i	-858993460	int
iNumberOfSheep	150	int
iWeightOfSheep	0x00000020b90fe0e0 {32757, 32662, 32439, 32391, 32209, 31...	int[1024]
[0]	32757	int
[1]	32662	int
[2]	32439	int
[3]	32391	int
[4]	32209	int
[5]	31673	int
[6]	31322	int
[7]	31115	int
[8]	31107	int
[9]	31101	int
[10]	30333	int
[11]	30191	int
[12]	30106	int
[13]	29658	int
[14]	29358	int
[15]	28745	int
[16]	28703	int
[17]	28253	int
[18]	28145	int
[19]	27753	int
[20]	27644	int
[21]	27529	int
[22]	27446	int
[23]	27350	int
[24]	26962	int
[25]	26924	int
[26]	26777	int
[27]	26500	int
[28]	26308	int
[29]	26299	int
[30]	26299	int
[31]	26299	int
[32]	26299	int
[33]	26299	int
[34]	26299	int
[35]	26299	int
[36]	26299	int
[37]	26299	int
[38]	26299	int
[39]	26299	int
[40]	26299	int
[41]	26299	int
[42]	26299	int
[43]	26299	int
[44]	26299	int
[45]	26299	int
[46]	26299	int
[47]	26299	int
[48]	26299	int
[49]	26299	int
[50]	26299	int
[51]	26299	int
[52]	26299	int
[53]	26299	int
[54]	26299	int
[55]	26299	int
[56]	26299	int
[57]	26299	int
[58]	26299	int
[59]	26299	int
[60]	26299	int
[61]	26299	int
[62]	26299	int
[63]	26299	int
[64]	26299	int
[65]	26299	int
[66]	26299	int
[67]	26299	int
[68]	26299	int
[69]	26299	int
[70]	26299	int
[71]	26299	int
[72]	26299	int
[73]	26299	int
[74]	26299	int
[75]	26299	int
[76]	26299	int
[77]	26299	int
[78]	26299	int
[79]	26299	int
[80]	26299	int
[81]	26299	int
[82]	26299	int
[83]	26299	int
[84]	26299	int
[85]	26299	int
[86]	26299	int
[87]	26299	int
[88]	26299	int
[89]	26299	int
[90]	26299	int
[91]	26299	int
[92]	26299	int
[93]	26299	int
[94]	26299	int
[95]	26299	int
[96]	26299	int
[97]	26299	int
[98]	26299	int
[99]	26299	int

自动窗口 局部变量 监视 1

分析总结

进一步，可以考虑使用二维数组存储羊的重量以及它们的编号，实现方式与上方代码高度类似。

第二题：高斯消去法

实验目的

题目略  
理解所给材料后将其写为代码

实验内容

分析

由题给材料，可以得出高斯消元法的实现过程如下（针对n阶方阵）：

```

1 //此处的u[][]是增广矩阵
2 for (int k = 0; k < n - 1; ++k) {
3     //列选主元
4     for (int i = k + 1; i < n; ++i) {
5         if (u[i][k] > u[k][k]) {
6             for (int j = 0; j < n + 1; ++j) {
7                 u[k][j] += u[i][j];
8                 u[i][j] = u[k][j] - u[i][j];
9                 u[k][j] -= u[i][j];
10            }
11        }
12    }
13
14    //归一化
15    for (int j = k + 1; j < n + 1; ++j) {
16        u[k][j] /= u[k][k];
17    }
18
19    //消去
20    for (int i = k + 1; i < n; ++i) {
21        for (int j = k + 1; j < n; ++j) {
22            u[i][j] -= u[i][k] * u[k][j];
23        }
24        u[i][n] -= u[i][k] * u[k][n];
25    }
26 }
27

```

实现思路与线性代数课程所讲的基本一致，通过矩阵的初等行变换将增广矩阵化为“阶梯型”。

进一步，得出解：

```

1 x[n-1] = u[n - 1][n] / u[n - 1][n - 1];
2 for (int i = n - 2; i >= 0; --i) {
3     double sum = 0;
4     for (int j = i + 1; j < n; ++j) {
5         sum += u[i][j] * x[j];
6     }
7     x[i] = u[i][n] - sum;
8 }

```

## 代码

```

1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 int main() {
6     double A[4][4] = { 1.1161, 0.1254, 0.1397, 0.1490, 0.1582, 1.1675, 0.1768,
7         0.1871, 0.2368, 0.2471, 0.2568, 1.2671, 0.1968, 0.2071, 1.2168, 0.2271 };

```

```

7   double b[4] = { 1.5471, 1.6471, 1.8471, 1.7471 };
8   double x[4] = { 0 };
9   double u[4][5];
10
11  cout.setf(ios::showpoint);
12  cout.flags(ios::fixed);
13  cout.precision(4);
14
15  cout << "MAT A =\n";
16  for (int i = 0; i < 4; ++i) {
17      for (int j = 0; j < 4; ++j) {
18          cout << setw(9) << A[i][j];
19      }
20      cout << endl;
21  }
22  cout << "MAT B =\n";
23  for (int j = 0; j < 4; ++j) {
24      cout << setw(9) << b[j];
25  }
26  cout << endl;
27
28  for (int i = 0; i < 4; ++i) {
29      for (int j = 0; j < 5; ++j) {
30          if (j == 4) u[i][j] = b[i];
31          else u[i][j] = A[i][j];
32      }
33  }
34
35  for (int k = 0; k < 3; ++k) {
36      //列选主元
37      for (int i = k + 1; i < 4; ++i) {
38          if (u[i][k] > u[k][k]) {
39              for (int j = 0; j < 5; ++j) {
40                  u[k][j] += u[i][j];
41                  u[i][j] = u[k][j] - u[i][j];
42                  u[k][j] -= u[i][j];
43              }
44          }
45      }
46      //归一化
47      for (int j = k + 1; j < 5; ++j) {
48          u[k][j] /= u[k][k];
49      }
50      //消去
51      for (int i = k + 1; i < 4; ++i) {
52          for (int j = k + 1; j < 4; ++j) {
53              u[i][j] -= u[i][k] * u[k][j];
54          }
55          u[i][4] -= u[i][k] * u[k][4];
56      }
57  }
58  cout << endl;
59

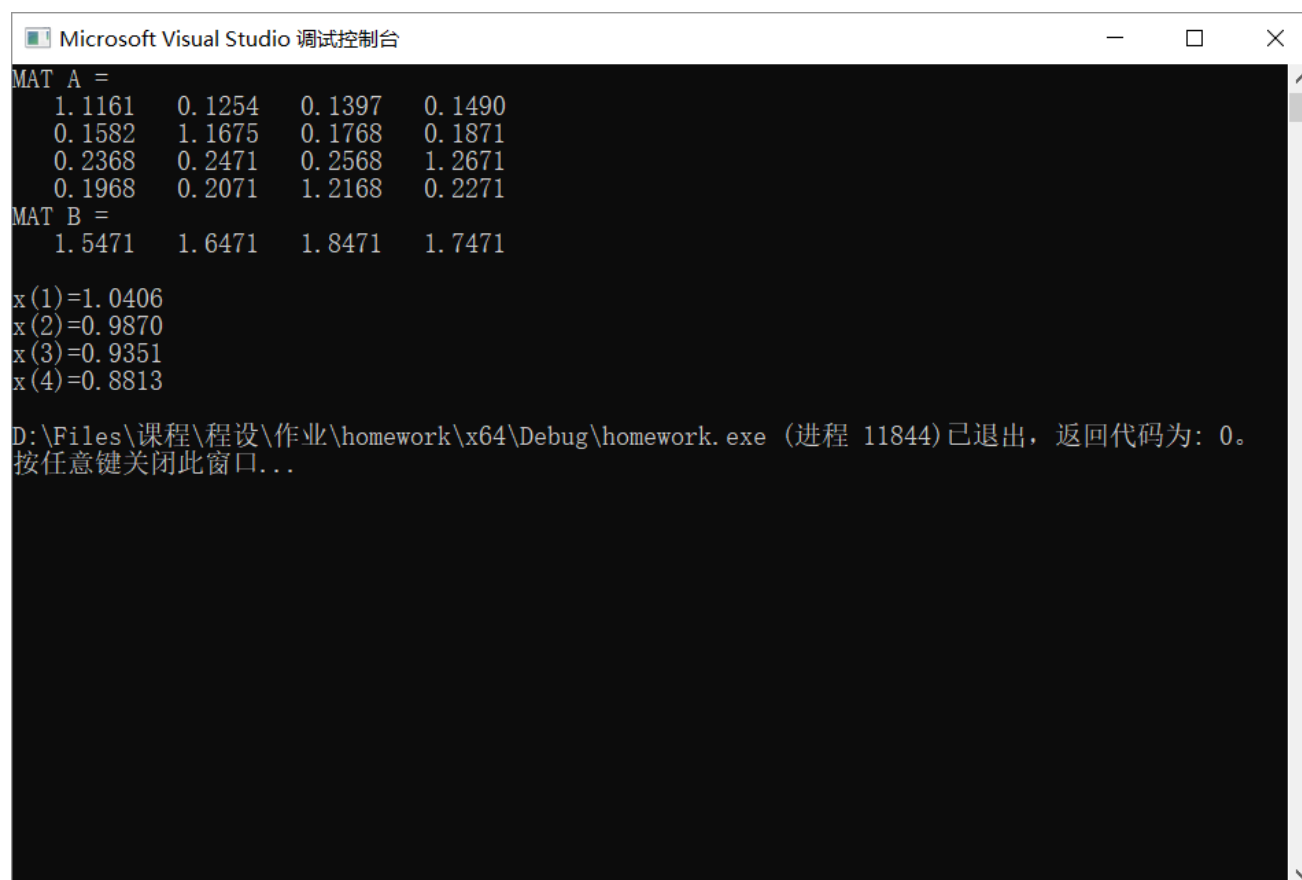
```

```

60     x[3] = u[3][4] / u[3][3];
61     for (int i = 2; i >= 0; --i) {
62         double sum = 0;
63         for (int j = i + 1; j < 4; ++j) {
64             sum += u[i][j] * x[j];
65         }
66         x[i] = u[i][4] - sum;
67     }
68     for (int i = 0; i < 4; ++i) {
69         cout << "x(" << i + 1 << ")=" << x[i] << endl;
70     }
71
72     return 0;
73 }

```

## 结果



```

Microsoft Visual Studio 调试控制台
MAT A =
1.1161  0.1254  0.1397  0.1490
0.1582  1.1675  0.1768  0.1871
0.2368  0.2471  0.2568  1.2671
0.1968  0.2071  1.2168  0.2271
MAT B =
1.5471  1.6471  1.8471  1.7471
x(1)=1.0406
x(2)=0.9870
x(3)=0.9351
x(4)=0.8813
D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 11844) 已退出，返回代码为：0。
按任意键关闭此窗口...

```

## 分析总结

这道题的难点在于，化为“阶梯型”的过程中，没有将对于求解用不到的数一并处理，造成处理完后与线性代数所讲的高斯消元法“形状不同”的效果。

## 第三题：鸡兔同笼

### 实验目的

某著名高校数学系教授在家教孙子做作业。题是这样：鸡和兔共15只，且有40只脚，问鸡和兔各几只？他开始给孙子解答，“设鸡的数量为，兔的数量为”.....还没等他讲完这些，旁边另一位小朋友已给出了了答案！他的算法是：假设鸡和兔都训练有素，吹一声哨，抬起一只脚， $40-15=25$ 。再吹哨，又抬起一只脚， $25-15=10$ 。此时鸡都一屁股坐地上了，兔子还两只脚立着。所以，兔子有 $10\div2=5$ 只，鸡有 $15-5=10$ 只。这种算法，让教授们情何以堪！试想一下：如果是计算机解题？又该如何编程呢？

## 实验内容

### 分析

最容易想到的就是遍历所有情况，找鸡兔数量满足要求的情况

但是可以应用题目中“另一位小朋友”的方法，先将 `腿数 - 头数*2`，进而得出兔子数量，然后推出鸡的数量。

### 代码

#### 遍历

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int iNumber, iFeet;
6      cout << "How many animals?\n";
7      cin >> iNumber;
8      cout << "And how many feet?\n";
9      cin >> iFeet;
10     for (int i = 0; i <= iNumber; ++i) {
11         if (4 * i + 2 * (iNumber - i) == iFeet) {
12             cout << "There are " << i << " rabbits and " << iNumber - i << "
chickens.\n";
13             return 0;
14         }
15     }
16     cout << "Input error\n";
17     return 0;
18 }
```

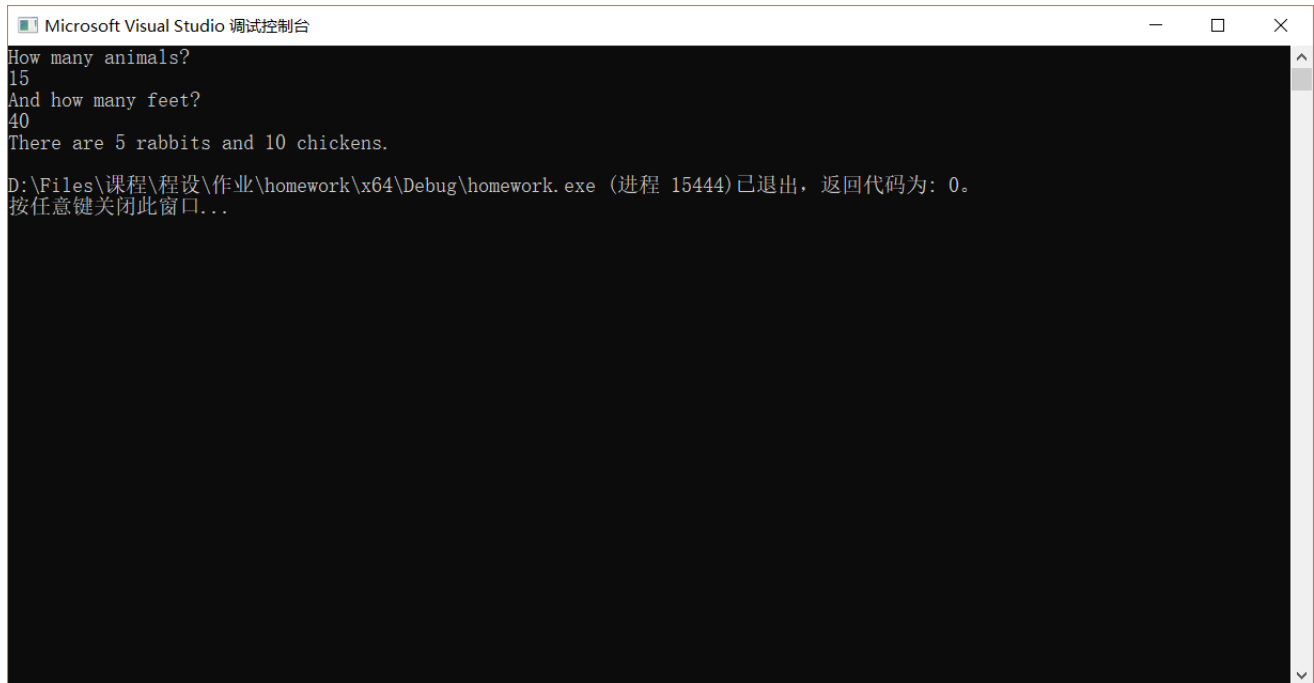
#### 小朋友方法

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int iNumber, iFeet;
6      cout << "How many animals?\n";
7      cin >> iNumber;
8      cout << "And how many feet?\n";
9      cin >> iFeet;
10
11     if ((iFeet - iNumber * 2) % 2 == 0) {
12         cout << "There are " << (iFeet - iNumber * 2) / 2 << " rabbits and " <<
iNumber - (iFeet - iNumber * 2) / 2 << " chickens.\n";
13     }
```



```
13     return 0;
14 }
15 cout << "Input error\n";
16
17 return 0;
18 }
```

## 结果



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio 调试控制台". The console output is as follows:

```
How many animals?
15
And how many feet?
40
There are 5 rabbits and 10 chickens.

D:\Files\课程\程设\作业\homework\x64\Debug\homework.exe (进程 15444) 已退出, 返回代码为: 0。
按任意键关闭此窗口...
```

## 分析总结

对于数字较小的情况下，两种方案的效率相差不多。但是当数字很大时，第二种方法的效率明显较好。