1 精度测试与控制

本实验假设所有代码文件放在同一文件夹中。请按照要求，产生可执行文件，并给出每一个步骤的截图。以下要求使用 Linux 或者 Windows 的 GCC 工具。

1.1 基础代码文件

以下包括.h 和.c 文件。

1.1.1 printBit.h 文件

```
void printIntBit(int, int); // 输出int型数据在内存中的0-1码
void printShortBit(short, int); // 输出short型在内存中的0-1码
void printCharBit(char, int); // 输出char型在内存中的0-1码
void printFloatBit(float, int); // 输出float型在内存中的0-1码
void printDoubleBit(double, int); // 输出double型在内存中的0-1码
```

1.1.2 printBit.c 文件

```
#include <stdio.h>
#include <stdlib.h>
#include "printBit.h"

void printIntBit(int i, int sizeOfDataType)
{
   printf("binary output:\t");
   int r = i;
   int *list = malloc(sizeof(int) * sizeOfDataType * 8);
   int j;
   int mask = 1;

   for (int j = 0; j < sizeOfDataType * 8; j++)
   {
      if (i & mask == 1)
      {
         list[j] = 1;
```

```
18          } else {
19              list[j] = 0;
20          }
21          i = i>>1;
22      }
23
24
25      for (int j = sizeOfDataType * 8 - 1; j >= 0; j--)
26      {
27          printf("%d", list[j]);
28          if(j % 4 == 0)
29              printf(" ");
30      }
31      // printf("\t decimal output: %d\t", r);
32
33      free(list);
34
35  }
36
37  void printShortBit(short i, int sizeOfDataType)
38  {
39      printf("binary output:\t");
40      short r = i;
41      int *list = malloc(sizeof(int) * sizeOfDataType * 8);
42      int j;
43      int mask = 1;
44
45
46      for (int j = 0; j < sizeOfDataType * 8; j++) {
47          if (i & mask == 1)
48          {
49              list[j] = 1;
50          } else {
51              list[j] = 0;
52          }
53          i = i>>1;
54      }
55
56
```

```
57    for (int j = sizeOfDataType * 8 - 1; j >= 0; j--)
58    {
59       printf("%d", list[j]);
60       if( j % 4 == 0)
61          printf(" ");
62    }
63    // printf("\t decimal output: %d\t", r);
64
65    free(list);
66
67 }
68
69 void printCharBit(char i, int sizeOfDataType)
70 {
71    printf("binary output:\t");
72    char r = i;
73    int *list = malloc(sizeof(int) * sizeOfDataType * 8);
74    int j;
75    int mask = 1;
76
77
78    for (int j = 0; j < sizeOfDataType * 8; j++)
79    {
80       if (i & mask == 1)
81       {
82          list[j] = 1;
83       }
84       else
85       {
86          list[j] = 0;
87       }
88       i = i>>1;
89    }
90
91
92    for (int j = sizeOfDataType * 8 - 1; j >= 0; j--)
93    {
94       printf("%d", list[j]);
95       if( j % 4 == 0)
```

```
96            printf(" ");
97        }
98        // printf("\t char output: '%c'\t", r);
99
100       free(list);
101
102   }
103
104   void printFloatBit(float k, int sizeOfDataType)
105   {
106       printf("binary output:\t");
107       float r = k;
108       long long *p = (long long *)&k;
109       long long i = *p;
110       int *list = malloc(sizeof(int) * sizeOfDataType * 8);
111       int j;
112       int mask = 1;
113
114
115       for (int j = 0; j < sizeOfDataType * 8; j++)
116       {
117           if (i & mask == 1)
118           {
119               list[j] = 1;
120           }
121           else
122           {
123               list[j] = 0;
124           }
125           i = i>>1;
126       }
127
128
129       for (int j = sizeOfDataType * 8 - 1; j >= 0; j--)
130       {
131           printf("%d", list[j]);
132           if( j % 4 == 0)
133               printf(" ");
134       }
```

```
135     // printf("\t decimal output (may be imprecise): %f\t", r);
136
137     free(list);
138
139 }
140
141 void printDoubleBit(double k, int sizeOfDataType) {
142     printf("binary output:\t");
143     float r = k;
144     long long *p = (long long *)&k;
145     long long i = *p;
146     int *list = malloc(sizeof(int) * sizeOfDataType * 8);
147     int j;
148     int mask = 1;
149
150
151     for (int j = 0; j < sizeOfDataType * 8; j++) {
152         if (i & mask == 1) {
153             list[j] = 1;
154         } else {
155             list[j] = 0;
156         }
157         i = i>>1;
158     }
159
160
161     for (int j = sizeOfDataType * 8 - 1; j >= 0; j--) {
162         printf("%d", list[j]);
163         if( j % 4 == 0)
164             printf(" ");
165     }
166     // printf("\t decimal output (may be imprecise): %lf\t", r);
167
168     free(list);
169
170 }
```

1.2 表示精度测试实验

以下包括代码和操作。

1.2.1 main1.c 文件

```
1  #include <stdio.h>
2  #include "printBit.h"
3
4  int main()
5  {
6     float f1 = 709394.3125;
7     printf("decimal input: %f\n", f1);
8     printFloatBit(f1, sizeof(f1));
9     printf("\n");
10    double d1 = 709394.3125;
11    printf("decimal input: %lf\n", d1);
12    printDoubleBit(d1, sizeof(d1));
13    printf("\n");
14    float f2 = 709394.15625;
15    printf("decimal input: %f\n", f2);
16    printFloatBit(f2, sizeof(f2));
17    printf("\n");
18    double d2 = 709394.15625;
19    printf("decimal input: %lf\n", d2);
20    printDoubleBit(d2, sizeof(d2));
21    printf("\n");
22    return 0;
23 }
```

1.2.2 运行

请给出运行截图。

1.2.3 回答问题

1. 第 7 行代码输出的数值与第 6 行中的数值一致吗？请针对第 10 至 11 行，第 14 至 15 行回答类似问题。

2. 若把第 15 行改为 `printf("decimal input: %.10f\n", f2);` 上述情况有何变化，有何不变？试简述背后的原因。

3. 仔细观察，第 8 行和第 12 行输出的二进制码有何关系？第 16 行和第 20 行之间有无这种关系？第 6 行和第 10 行代码中的数字，和第 14 行和第 18 行代码中的数字相比，哪一个的有效数位多？

4. 以上分析对实际编程有何指导意义？

1.3 表示与运算精度测试实验

以下包括代码和操作。

1.3.1 main2.c 文件

```c
#include <stdio.h>
#include <math.h> // 支持绝对值运算
#include "printBit.h"
#define EPS 0.00001 // 精度
int main()
{
   double d1 = 123456789.9 * 9;
   printf("decimal input: %.7lf\n", d1);
   printDoubleBit(d1, sizeof(d1));
   printf("\n");
   double d2 = 1111111109.1; //
   printf("decimal input: %.7lf\n", d2);
   printDoubleBit(d2, sizeof(d2));
   printf("\n");
   printf("d1 == d2: %d\n", d1 == d2); //
   printf("d1 == d2: %d\n", fabs(d1 - d2) <= EPS);
   return 0;
}
```

1.3.2 运行

请给出运行截图。

1.3.3 回答问题

1. 求第 7 行中的数学式手工计算得到的数值。它与第 8 行代码输出的数值一致吗？请针对第 11 至 12 行回答类似问题。
2. 第 9 行和第 13 行输出的 0-1 序列相同吗？如果不同，请指出差别。
3. 从编程的角度看，第 15 行说明了什么？
4. 从编程的角度看，第 16 行说明了什么？
5. 以上分析对编程有何指导意义？

1.4 精度控制实验

以下包括代码和操作。

1.4.1 `main3.c` 文件

```c
#include <stdio.h>
#include <math.h>
#include "printBit.h"
#define EPS1 0.00000000001 // 10^(-11)
#define EPS2 0.000000000001 // 10^(-12)
int main()
{
    float f1 = 0.709394709394; //
    printf("decimal input: %.9f\n", f1);
    printFloatBit(f1, sizeof(f1));
    printf("\n");
    float f2 = 0.709394709395;
    printf("decimal input: %.9f\n", f2);
    printFloatBit(f2, sizeof(f2));
    printf("\n");
    double g1 = 0.709394709394; //
    printf("decimal input (shown in 10 significant digits):
        %.10lf\n", g1);
    printf("decimal input (shown in 11 signifcant digits):
        %.11lf\n", g1);
    printDoubleBit(g1, sizeof(g1));
```

```
20    printf("\n");
21    double g2 = 0.709394709395;
22    printf("decimal input (shown in 10 signifcant digits):
          %.10lf\n", g1);
23    printf("decimal input (shown in 11 signifcant digits):
          %.11lf\n", g2);
24    printDoubleBit(g2, sizeof(g2));
25    printf("\n");
26    printf("Comparisons at given precision EPS1 equal to
          10^(-11):\n");
27    printf("With type float:\n");
28    if(fabs(f1 - f2) < EPS1) //
29       printf("%.12f == %.12f\n", f1, f2);
30    else
31       printf("%.12f != %.12f\n", f1, f2);
32    printf("With type double:\n"); //
33    if(fabs(g1 - g2) < EPS1) //
34       printf("%.12lf == %.12lf\n", g1, g2);
35    else
36       printf("%.12lf != %.12lf\n", g1, g2);
37    printf("Comparisons at given precision EPS2 equal to
          10^(-12):\n");
38    printf("With type float:\n");
39    if(fabs(f1 - f2) < EPS2) //
40       printf("%.12f == %.12f\n", f1, f2);
41    else
42       printf("%.12f != %.12f\n", f1, f2);
43    printf("With type double:\n"); //
44    if(fabs(g1 - g2) < EPS2) //
45       printf("%.12lf == %.12lf\n", g1, g2);
46    else
47       printf("%.12lf != %.12lf\n", g1, g2);
48 }
```

1.4.2 运行

请给出运行截图。

1.4.3 回答问题

1. 第 9 行代码输出的数值与第 8 行中的数值一致吗？请针对第 12 至 13 行回答类似问题。
2. 第 17 和 18 行各自输出的结果，与第 16 行中的数字是否一致？请针对第 21、22 和 23 行回答类似的问题。简述背后的原因。
3. 第 9 和 13 行输出的内容相同吗？若不同，指出区别所在。
4. 第 10 和 14 行输出的 0-1 码相同吗？若不同，指出区别所在。
5. 第 19 和 24 行输出的 0-1 码相同吗？若不同，指出区别所在。
6. 简述上面 3 个问题展现的现象的背后原因。
7. 考虑第 28、33、39 和 44 行中的判断，它们各自成立吗？简述背后的原因。

1.5 涉及运算的精度控制实验

以下包括代码和操作。

1.5.1 `main4.c` 文件

```c
#include <stdio.h>
#include <math.h>

#include "printBit.h"

#define EPS1 0.0000001 // 10^(-7)
#define EPS2 0.000001 // 10^(-6)

int main()
{
   double d1 = 123456789.9 * 9;
   printf("decimal input: %.7lf\n", d1);
   printDoubleBit(d1, sizeof(d1));
   printf("\n");
   double d2 = 1111111109.1; //
```

```
16      printf("decimal input: %.7lf\n", d1);
17      printDoubleBit(d2, sizeof(d2));
18      printf("\n");
19      printf("d1 == d2: %d\n", d1 == d2); //
20      printf("Comparions at given precision 10^(-7):\n");
21      if(fabs(d1 - d2) < EPS1)
22      {
23          printf("%.7lf == %.7lf\n", d1, d2);
24      }
25      else
26      {
27          printf("%.7lf != %.7lf\n", d1, d2);
28      }
29      printf("Comparions at given precision 10^(-6):\n");
30      if(fabs(d1 - d2) < EPS2)
31      {
32          printf("%.7lf == %.7lf\n", d1, d2);
33      }
34      else
35      {
36          printf("%.7lf != %.7lf\n", d1, d2);
37      }
38      return 0;
39  }
```

1.5.2 运行

请给出运行截图。

1.5.3 回答问题

1. 考虑第 21 和 30 行中的判断，它们各自成立吗？简述背后的原因。

1.6 浮点型文字的精度

以下包括代码和操作。

1.6.1 main5.c 文件

```c
#include <stdio.h>
int main()
{
   float f = 34.6;
   printf("f == 34.6 returns %d\n", f == 34.6);
   double d = 34.6;
   printf("d == 34.6 returns %d\n", d == 34.6);
   return 0;
}
```

1.6.2 运行

请给出运行截图。

1.6.3 回答问题

1. 使用浮点型的变量 f 表示 34.6 时，用 == 判断二者严格相等得到的结果是什么?
2. 使用双精度的变量 d 表示 34.6 时，用 == 判断二者严格相等得到的结果是什么?
3. 简述其中的原因。

1.7 运算中的近似

以下包括代码和操作。

1.7.1 main6.c 文件

```c
#include <stdio.h>
#include "printBit.h"

int main()
{
   double a = 1.2;
```

```
    printf("decimal input: %lf\n", a);
    printDoubleBit(a, sizeof(a));
    printf("\n");
    double b = 2.5;
    printf("decimal input: %lf\n", b);
    printDoubleBit(b, sizeof(b));
    printf("\n");
    double c = 3.0;
    printf("decimal input: %lf\n", c);
    printDoubleBit(c, sizeof(c));
    printf("\n");
    double product = a * b;
    printf("decimal input: %lf\n", product);
    printDoubleBit(product, sizeof(product));
    printf("\n");
    return 0; //
}
```

1.7.2 运行

请给出运行截图。

1.7.3 回答问题

1. 手工计算 a 与 b 的乘积。
2. 变量 a 在内存中的二进制码是什么？这是 1.2 的精确表示吗？变量 b 和 c 呢？
3. 变量 product 在内存中的二进制码是什么？它和变量 c 的二进制码有什么等量和不等关系？
4. 变量 product 的二进制码，是 3.0 的精确表示吗？
5. 简述以上实验步骤说明什么问题。

1.8 实验报告写作要求

1. 步骤详细；

2. 表述简明；

3. 图文并茂；

4. 逻辑流畅。