# 1 数据类型测试

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 整数类型实验

以下包括代码和操作。

### 1.2.1 main1.c 文件

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

int main()
{
    short m1 = 1234;
    printf("decimal input: %d\n", m1);
    printShortBit(m1, sizeof(m1)); // 输出m1在内存中的0-1码
    printf("\n");
    short m2 = -1234;
    printf("decimal input: %d\n", m2);
    printShortBit(m2, sizeof(m2)); // 输出m2在内存中的0-1码
    printf("\n");
    int n1 = 709394;
    printf("decimal input: %d\n", n1);
    printIntBit(n1, sizeof(n1)); // 输出n1在内存中的0-1码
    printf("\n");
    int n2 = -709394;
    printf("decimal input: %d\n", n2);
    printIntBit(n2, sizeof(n2)); // 输出n2在内存中的0-1码
    printf("\n");
    return 0;
}
```

### 1.2.2 运行

请给出运行截图。

### 1.2.3 回答问题

1. 考虑上述 C 语言代码，第 7 行代码输出的数值与第 6 行中的数值一致吗？第 8 行输出什么？它与第 6 行代码中的数是什么关系？请简要说

明理由。

2. 请针对第 10 至 12 行，回答与上述相应的问题。
3. 请针对第 14 至 16 行，回答与上述相应的问题。
4. 请针对第 18 至 20 行，回答与上述相应的问题。

## 1.3 `float` 型实验

以下包括代码和操作。

### 1.3.1 `main2.c` 文件

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

int main()
{
    float f1 = 34.0;
    printf("decimal input: %f", f1);
    printFloatBit(f1, sizeof(f1));
    printf("\n");
    float f2 = -34.0;
    printf("decimal input: %f", f2);
    printFloatBit(f2, sizeof(f2));
    printf("\n");
    float f3 = 34.5;
    printf("decimal input: %f", f3);
    printFloatBit(f3, sizeof(f3));
    printf("\n");
    float f4 = -34.5;
    printf("decimal input: %f", f4);
    printFloatBit(f4, sizeof(f4));
    printf("\n");
    float f5 = 34.6;
    printf("decimal input: %f", f5);
    printFloatBit(f5, sizeof(f5));
    printf("\n");
```

```
26    float f6 = -34.6;
27    printf("decimal input: %f", f6);
28    printFloatBit(f6, sizeof(f6));
29    printf("\n");
30    return 0;
31 }
```

### 1.3.2 运行

请给出运行截图。

### 1.3.3 回答问题

1. 考虑上述 C 语言代码，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，第 18 至 19 行，第 22 至 23 行，第 26 至 27 行回答类似问题。
2. 第 8 行和第 12 行输出的二进制码有何关系？请针对第 16 行和第 20 行，以及第 24 行和第 28 行，回答类似问题。
3. 如果把三个正浮点数输出的二进制码归为一类，三个负浮点数输出的二进制码归为另一类，请给出一个简单的区分标准。

## 1.4 `double` 型实验

以下包括代码和操作。

### 1.4.1 `main3.c` 文件

```
1  #include <stdio.h>
2  #include "printBit.h"
3
4  int main()
5  {
6     double d1 = 34.0;
7     printf("decimal input: %17lf\n", d1);
8     printDoubleBit(d1, sizeof(d1));
9     printf("\n");
```

```
10      double d2 = -34.0;
11      printf("decimal input: %17lf\n", d2);
12      printDoubleBit(d2, sizeof(d2));
13      printf("\n");
14      double d3 = 34.5;
15      printf("decimal input: %17lf\n", d3);
16      printDoubleBit(d3, sizeof(d3));
17      printf("\n");
18      double d4 = -34.5;
19      printf("decimal input: %17lf\n", d4);
20      printDoubleBit(d4, sizeof(d4));
21      printf("\n");
22      double d5 = 34.6;
23      printf("decimal input: %.17lf\n", d5);
24      printDoubleBit(d5, sizeof(d5));
25      printf("\n");
26      double d6 = -34.6;
27      printf("decimal input: %.17lf\n", d6);
28      printDoubleBit(d6, sizeof(d6));
29      printf("\n");
30      return 0;
31  }
```

### 1.4.2 运行

请给出运行截图。

### 1.4.3 回答问题

1. 考虑上述 C 语言代码，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，第 18 至 19 行，第 22 至 23 行，第 26 至 27 行回答类似问题。

2. 第 8 行和第 12 行输出的二进制码有何关系？请针对第 16 行和第 20 行，以及第 24 行和第 28 行，回答类似问题。

3. 如果把三个正浮点数输出的二进制码归为一类，三个负浮点数输出的二进制码归为另一类，请给出一个简单的区分标准。

## 1.5 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。