1 随机数测试

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 用 time() 函数获取种子实验的代码文件

以下包括代码和操作。

1.1.1 wait.h 文件

```
void wait(long long);
```

1.1.2 wait.c 文件

```
void wait(long long tick_num)
{
   while(tick_num--){;}
}
```

1.1.3 main1.c 文件

```
#include <stdio.h>
#include <time.h> // 支持time()函数
#include <stdlib.h> // 支持伪随机数操作
#include "wait.h"
int main()
{
   printf("RAND_MAX: %d\n", RAND_MAX);
   int sequence_num = 8; // 随机序列的个数
   int sequence_length = 5; // 随机序列的长度
   long long tick_n = 10000000; // 延时的节拍数
   for(int i = 0; i < sequence_num; i++) //
       循环sequence_num次，从而产生多个序列
   {
```

```
13        // 每个序列重新设置种子，希望每次从不同的序列中获取数据
14        int seed = time(NULL); // 通过time()函数来获取种子
15        srand(seed); // 设置种子
16        for(int j = 0; j < sequence_length; j++) //
              循环sequence_length次，从而产生多个随机数
17        {
18            int d; // 用于保存生成的伪随机数
19            d = rand(); // 获得下一个伪随机数
20            printf("%d\t", d); // 输出
21        }
22        printf("\n"); // 回车换行
23        wait(tick_n); // 延时等待
24    }
25    return 0;
26 }
```

1.1.4 步骤

1. 根据以上代码产生可执行文件。
2. 运行上述可执行文件 4 次，每次分别把第 11 行中的数值设为 `10000000`（即 10^7），`100000000`（即 10^8），`1000000000`（即 10^9）和 `10000000000`（即 10^{10}）输入到 `tick_n`，并记录输出结果。
3. 分析你得到的实验结果，并简要叙述你的实验和分析对编程的指导意义。

1.2 人工指定种子的代码文件

以下包括代码和操作。

1.2.1 `main2.c` 文件

```
1  #include <stdio.h>
2  #include <time.h> // 支持time()函数
3  #include <stdlib.h> // 支持伪随机数操作
4  int main()
5  {
```

```
    printf("RAND_MAX: %d\n", RAND_MAX);
    int sequence_num = 8; // 随机序列的个数
    int sequence_length = 5; // 随机序列的长度
    for(int i = 0; i < sequence_num; i++) //
        循环sequence_num次，从而产生多个序列
    {
        // 每个序列重新设置种子，希望每次从不同的序列中获取数据
        int seed = i + 1; // 人工指定种子
        srand(seed); // 设置种子
        for(int j = 0; j < sequence_length; j++) //
            循环sequence_length次，从而产生多个随机数
        {
            int d; // 用于保存生成的伪随机数
            d = rand(); // 获得下一个伪随机数
            printf("%d\t", d); // 输出
        }
        printf("\n");
    }
    return 0;
}
```

1. 根据以上代码产生可执行文件。
2. 运行上述可执行文件 4 次，并记录输出结果。
3. 分析你得到的实验结果，并简要叙述你的实验和分析对编程的指导意义。

1.3 均匀性测试

以下包括代码和操作。

1.3.1 main3.c 文件

```
#include <stdio.h>
#include <time.h> // 支持time()函数
#include <stdlib.h> // 支持伪随机数操作

```

```
5  int main()
6  {
7      srand(time(NULL)); // 布置种子
8      printf("RAND_MAX: %d\n", RAND_MAX);
9      int sequence_length = 10; // 随机序列的长度
10     for(int i = 0; i < sequence_length; i++) //
           循环sequence_length次，从而产生多个随机数
11     {
12         int d; // 用于保存生成的伪随机数
13         d = rand() % 10; // 获得下一个0-9之间的伪随机数
14         printf("%d\t", d); // 输出
15     }
16     return 0;
17 }
```

1. 根据以上代码产生可执行文件，并把产生的文件命名为 `main3.exe`。
2. 以下操作把输出结果保存在 `res.txt` 文件中，并统计 `0-9` 各个数字出现的频率。
   (a) 检查 `main3.exe` 所在文件夹是否存在 `res.txt` 文件，如果有，请删掉。
   (b) 运行 `main3.exe » res.txt`。
   (c) 把 `res.txt` 的内容复制到 Microsoft Word 或者 WPS 文字编辑工具中，统计 `0-9` 各个数字出现的频数，然后计算相应的频率。(频率等于频数除以总数。)
3. 把第 9 行的 `10` 先后改为 `100`，`1000`，`10000`，重新统计 `0-9` 各个数字出现的频率。
4. 从以上实验步骤的结果中，你得到什么结论?

1.4 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。