C 语言的随机函数

- int rand(void)　　产生从 0 到 RAND_MAX(32767)之间的随机数

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{   int i;
    for(i = 0; i < 10; i++) //打印出 10 个随机数
        printf("%d\n", rand() );
}
```

编译运行，发现的确产生随机数了。但是每次运行程序产生的随机数序列都是一样的，为什么呢？因为 rand()函数是根据一个种子数字 seed number（如开机时间）为基准，按照一系列复杂公式计算出来的，本质上是“伪随机数”。为了改变这个种子数字的值，C 语言提供了 srand()函数。

- void srand(unsigned int seed)　　为 rand()函数生成的伪随机数序列设置起点种子值

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>          //使用当前时钟作为种子
#define RANDMAX 100   //随机整数范围的最大值
#define RANDMIN 1          //随机整数范围的最小值
int main(void)
{
    int i;
    srand((unsigned)time(NULL));  //初始化随机数
    printf("随机数序列为：\n");
    for(i = 0; i < 10; i++)              //打印出 10 个随机数
        printf("%d\n", rand() % (RANDMAX - RANDMIN + 1) + RANDMIN);
}
```

再运行程序，会发现每次产生的随机数都不一样了。这是因为这里采用了当前运行时间作为随机种子数，而前后两次运行时间并不相同,所以就产生了符合要求的"随机"的随机数了。所以，要想产生不同的随机数序列，在使用 rand()之前需要先调用 srand()。

还可以在隔一段运行时间后使用 srand()提供一个新的种子数，从而进一步“随机化”rand()的输出结果。

接收字符输入

```
/*输入单个或多个字符时，可能会接收到之前输入缓冲区内遗留下来的垃圾内容*/
#include <stdio.h>
int main() {
```

```
    double x;
    char str[20], ch;
    printf("Input a real number: ");
    scanf("%lf", &x);  //试试输入 1.414、3.1o4
    printf("x = %f\n", x);
    scanf("%c", &ch);
    printf("ch(ASCII) = %d\n", ch);

    printf("Input a string: ");
    scanf("%s", str);
    printf("your string is \"%s\"\n", str);
    ch = getchar();
    printf("ch(ASCII) = %d\n", ch);
}
```

因此在接收字符输入之前总是运行 fflush(stdin);可以把之前输入缓冲区内遗留下来的换行符等多余的清除掉。

```
scanf("%lf", &x);
do{
    fflush(stdin);
    printf("按 W 键产生你的赌盘数字：\n");
    chW = getchar();
}while(chW!='W' && chW!='w');
```

绘制流程图

使用你熟悉的绘图软件绘制流程图。如可视化快速算法原型工具 Raptor，或者微软 office 套件中的 visio，或者 wps 应用中的流程图工具绘制。