2020-05-05

第2章 循环结构程序设计

用计时函数测试程序效率

```
 #include <time.h>

    printf("Time used = %.2f\n",(double)clock()/CLOCKS_PER_SEC);
```

输入输出重定向

- 基础版本

```
freopen("input.txt","r",stdin);
freopen("output.txt","w",stdout);
```

- 本地版本

```
#define LOCAL

#ifdef LOCAL

    freopen("input.txt","r",stdin);
    freopen("output.txt","w",stdout);

#endif
```

- fopen 版本

```
FILE *fin,*fout;
fin = fopen("data.in","rb");
fout = fopen("data.out","wb");
//输入
fscanf(fin,"%d",&x);
//输出
fprintf(fout,"%d %d %.3f\n",min,max,(double)s/n);
//关闭文件
fclose(fin);
fclose(fout);
```

第3章 数组和字符串

数组

- 数组赋值函数 memcpy

```
#include<string.h>

memcpy(b,a,sizeof(int)*k);//int型
memcpy(b,a,sizeof(double)*k);//double型
memcpy(b,a,sizeof(a));//全部复制
```

- 数组清零

```
#include<string.h>
memset(a,0,sizeof(a));
```

- 蛇形填数(伪代码)

```
#define maxn 20
int a[maxn][maxn],tot;
tot = a[x=0][y=n-1] =1;
while(tot< n*n)
{
    while(x+1<n && !a[x+1][y])  a[++x][y] = ++tot;
    while(y-1>=0 && !a[x][y-1]) a[x][--y] = ++tot;
    while(x-1>=0 && !a[x-1][y]) a[--x][y] = ++tot;
    while(y+1<n && !a[x][y+1])  a[x][++y] = ++tot;
}
```

- 竖式问题

```
#include<stdio.h>
#include<string.h>
int main()
{
    int count = 0;
    char s[20]; buf[99];
    scanf("%s",s);
    for(int abc=111; abc <= 999; abc++)
        for(int de =11; de <= 99; de++)
        {
            int x = abc*(de%10), y = abc*(de/10), z = abc*de;
            sprintf(buf,"%d%d%d%d%d%d"), abc, de, x, y, z);
            int ok = 1;
            for(int i = 0; i <strlen(buf); i++)
                if(strchr(s, buf[i]) == NULL) ok = 0;
            if(ok)
            {
                printf("<%d>\n", ++count);
                printf("%5d\nX%4d\n-----\n%5d\n%4d\n-----\n%5d\n\n",
abc,de,x,y,z);
            }
```

```
        }
    printf("The number of solutions = %d\n", count);
    return 0;
}
```

竞赛题目选讲

- TeX中的引号$UVa272$

```
#include<stdio.h>
int main(){
    int c,q=1;
    while((c=getchar()) != EOF){
        if(c = '"') { printf("%s", q ? "``" ; "''"); q = !q; }
        else printg("%c", c);
    }
    return 0;
}
```

善用常量数组往往能简化代码。定义常量数组时无须指明大小，编译器会计算。

```
#include<stdio.h>
char s[] = "`1234567890-=QWERTYUIOP[]\\ASDFGHJKL;'ZXCVBNM,./";
int main(){
    int i, c;
    while((c = getchar()) != EOF){
        for (i=1; s[i] && s[i]!=c;i++);
        if(s[i]) putchar(s[i-1]);
        else putchar(c);
    }
    return 0;
}
```

回文词$UVa401$

输入一个字符串，判断它是否为回文串以及镜像串。输入字符串保证不含数字0。（使用常量数组）

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
const char* rev = "A   3  HIL JM O   2TUVWXY51SE Z  8 ";
const char* msg[] = {"not a palindrome", "a regular palindrome", "a mirrored
string", "a mirrored palindrome"};

char r(char ch) {
  if(isalpha(ch)) return rev[ch - 'A'];
  return rev[ch - '0' + 25];
}
```

```
int main() {
  char s[30];
  while(scanf("%s", s) == 1) {
    int len = strlen(s);
    int p = 1, m = 1;
    for(int i = 0; i < (len+1)/2; i++) {
      if(s[i] != s[len-1-i]) p = 0;
      if(r(s[i]) != s[len-1-i]) m = 0;
    }
    printf("%s -- is %s.\n\n", s, msg[m*2+p]);
  }
  return 0;
}
```

猜数字游戏的提示$UVa340$)

直接统计可得A为了求B对于每个数字（1~9），统计二者出现次数的最小值之和就是该数字对B的贡献。最后输出AB-A即可。

```
#include<stdio.h>
#define maxn 1000 + 10

int main() {
  int n, a[maxn], b[maxn];
  int kase = 0;
  while(scanf("%d", &n) == 1 && n) { // n=0时输入结束
    printf("Game %d:\n", ++kase);
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    for(;;) {
      int A = 0, B = 0;
      for(int i = 0; i < n; i++) {
        scanf("%d", &b[i]);
        if(a[i] == b[i]) A++;
      }
      if(b[0] == 0) break; //正常的猜测序列不会有0，所以只判断第一个数是否为0即可
      for(int d = 1; d <= 9; d++) {
        int c1 = 0, c2 = 0; //统计数字d在答案序列和猜测序列中各出现多少次
        for(int i = 0; i < n; i++) {
          if(a[i] == d) c1++;
          if(b[i] == d) c2++;
        }
        if(c1 < c2) B += c1; else B += c2;
      }
      printf("    (%d,%d)\n", A, B-A);
    }
  }
  return 0;
}
```

生成元$UVa1583$ 预处理加散列表。比较简单，代码略。

环状序列$UVa1584$ 用 ans 表示目前为止，字典序最小串在输入串中的起始位置，然后不断更新ans一个值得注意的地方是，对于环形序列，采用他对字符串长度的mod值来表示它的相对位置。

```
#include<stdio.h>
#include<string.h>
#define maxn 105

//环状串s的表示法p是否比表示法q的字典序小
int less(const char* s, int p, int q) {
  int n = strlen(s);
  for(int i = 0; i < n; i++)
    if(s[(p+i)%n] != s[(q+i)%n])
      return s[(p+i)%n] < s[(q+i)%n];
  return 0; //相等
}

int main() {
  int T;
  char s[maxn];
  scanf("%d", &T);
  while(T--) {
    scanf("%s", s);
    int ans = 0;
    int n = strlen(s);
    for(int i = 1; i < n; i++)
      if(less(s, i, ans)) ans = i;
    for(int i = 0; i < n; i++)
      putchar(s[(i+ans)%n]);
    putchar('\n');
  }
  return 0;
}
```

注解与习题

用ASCII 编码表示字符

进制转换与移位运算符

补码表示法

在大多数的计算机内部，整数采用的是补码表示法。

思考题

必要的存储量：有些时候不一定要开数组即可完成任务。

黑盒测试

算法竞赛一般采用黑盒测试。

超时的原因：效率低；读取数据错误；程序死循环；程序崩溃但退出异常。

在线评测系统 $Online Judge,OJ$

西班牙$Valladolid$大学的$UVaOJ$

第四章 函数和递归

自定义函数和结构体

- 采用$struct$ 的写法

```
struct Point{ double x,y;};
double dist(struct Point a,struct Point b)
{
    return hypot(a.x-b.x, a.y-b.y);
}
```

- 采用$typedef$的写法

```
typedef struct{ double x,y;}Point;
double dist(Point a,Point b)
{
    return hypot(a.x-b.x, a.y-b.y);
}
```

素数判定

`is_prime` 的参数并不是`long long`型，因为在n很大时，这个函数并不能很快计算出结果。

```
int is_prime(int n)
{
    if (n <= 1) return 0;
    int m = floor(sqrt(n) + 0.5);
    for (int i = 2; i<= m; i++)
        if(n % i == 0) return 0;
    return 1;
}
```