A 字符串替换.改

时间限制：1000ms 内存限制：65536kb

通过率：1437/1509 (95.23%) 正确率：1437/2961 (48.53%)

题目描述

从标准输入上读入的数据中，每行中最多包含一个字符串“_xy_”且除了字符串“_xy_”外输入数据中不包括下划线字符，将输入行中的“_xy_”替换为“_ab_”,在标准输出上输出替换后的结果

输入

多组数据输入，每组数据为一行字符串，要求如题目描述。

输出

输出替换后的字符串，若没有进行过替换则输出原字符串。

输入样例

```
xy_xy_2018-4-27
```

输出样例

```
xy_ab_2018-4-27
```

HINT

使用函数 strchr()和/或 strrchr()

考察知识点

返回指针的函数

难度系数 1

解题思路

本题可参照书 P150 例 7-4 字符串替换中代码，将输入保存在 buf 中，并利用 strstr 函数查找”_xy_”位置 p，将输出字符串分为三部分：”_xy_”前内容（buf 开始直至 p-1 位置），”_xy_”替换为”_ab_”，”_xy_”后内容（p+”_xy_”长度开始，直至字符串结束）；但题目要求稍微变化了一些，也可以使用 strchr 函数定位’_’字符。

解法一：

```
#include <stdio.h>
#include <string.h>

int main()
{
	char buf[BUFSIZ], *p, *str = "_xy_";

	while (scanf("%s", buf) != EOF)
	{
		p = strstr(buf, str);
		if (p == NULL)
		{
			printf("%s\n", buf);
			continue;
		}
		*p = '\0';
		printf("%s_ab_%s\n", buf, p + strlen(str));
	}
	return 0;
}
```

解法二：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
	char buf[BUFSIZ],*p;
	while(fgets(buf,BUFSIZ,stdin)!=NULL)
	{
		p=strchr(buf,'_');
		if(p==NULL)
		{
			printf("%s",buf);
			continue;
		}
		*p='\0';
		printf("%s_ab_%s",buf,p+4);
	 }
	return 0;

}
```

第几月.改

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 1481 总提交人数: 1531

题目描述

计算并输出非闰年的第 x 天是第几月。

输入

一个整数 x$(1\leq x\leq 365)$。

输出

英文单词 "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"

输入样例 1

```
1
```

输出样例 1

```
January
```

输入样例 2

```
60
```

输出样例 2

```
March
```

输入样例 3

```
90
```

输出样例 3

```
March
```

考察知识点：指针数组
解题思路：先用一个指针数组将各个月份的英文初始化，并提前求出各个月份的长度。这样根据 x 便可以求得是第几月，然后通过访问指针数组来输出。

代码：

```
#include<stdio.h>

char *month[] = {"January", "February", "March", "April",

                 "May", "June", "July", "August",

                 "September", "October", "November", "December"};

int month_days[] = {31, 28, 31, 30, 31, 30,

                    31, 31, 30, 31, 30, 31};

int main() {

    int i, x;

    scanf("%d", &x);

    for (i = 0; i < 12; i++)

     if ((x -= month_days[i]) <= 0)

         break;

    printf("%s\n", month[i]);

    return 0;

}
```

C 单词集合

时间限制：1000ms 内存限制：65536kb

通过率：152/418 (36.36%) 正确率：152/1345 (11.30%)

题目描述

有两个以花括号定义的单词表，每个表中单词的个数不大于 100100，每个单词的长度不超过 3636 个字符，单词间以逗号分隔，单词前后可能有空白符。

如果两个单词表中有相同的单词，则把它们按照升序输出（多个相同单词只输出一次），单词之间以一个空格符分隔，否则，则输出"NONE"。

输入

一组输入，包含两对花括号和其中的单词，单词中只会出现小写字母。

输出

一行，相同的单词

输入样例

```
{an, apple, tree}

{a, red, apple}
```

输出样例

```
apple
```

输入样例 2

```
{as, pos, tag, mid, less}

{tag, as, less, two, three}
```

输出样例 2

```
as less tag
```

考察知识点

字符数组排序，指针应用

难度系数 3

解题思路

用字符串指针数组保存多个字符串，在交换时只需交换指针，可以增加排序效率。可分为如下步骤：

1. 建立两个指针数组 strArray1,strArray2 分别保存两个集合中的单词；
2. 用 malloc 函数请求分配保存一个字符串，字符串首地址记为 tmp；
3. 将 tmp 按升序插入 strArray 中，如果已经存在则不插入；
4. 重复 2 和 3，直到全部单词存入数组；
5. 按顺序比较两个数组中的单词，如果有相同的则输出；
6. 如果不存在相同单词，输出 NONE；
7. 对于 malloc 分配的内存，用完后要 free。

AC 代码：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define WORD_NUMBER 100
#define WORD_LENGTH 36

int ReadWord(char* strArray[]);

int main()
{
    char *strArray1[WORD_NUMBER], *strArray2[WORD_NUMBER];
    int arrayLength1 = 0, arrayLength2 = 0, i, j, ret, flag = 1;
    arrayLength1 = ReadWord(strArray1);
    arrayLength2 = ReadWord(strArray2);
    for (i = 0, j = 0; i < arrayLength1 && j < arrayLength2;)
    {
        ret = strcmp(strArray1[i], strArray2[j]);
        if (ret)
        {
            if (ret > 0)
            {
                j++;
            }
            else
            {
                i++;
            }
```

```
        }
        else
        {
            printf("%s ", strArray1[i]);
            i++;
            j++;
            flag = 0;
        }
    }
    if (flag)
    {
        printf("NONE");
    }
    for (i = 0; i < arrayLength1; i++)
    {
        free(strArray1[i]);
    }
    for (i = 0; i < arrayLength2; i++)
    {
        free(strArray2[i]);
    }
    return 0;
}

int ReadWord(char* strArray[])
{
    int num = 0, i, j, res;
    char *tmp = (char*)malloc((WORD_LENGTH + 1) * sizeof(char));
    scanf("%*[^{]");
    scanf("%*[{ ]");
    while (scanf("%[^ ,}]%*[, ]", tmp) == 1)
    {
        for (i = 0; i < num; i++)
        {
            if ((res = strcmp(tmp, strArray[i])) <= 0)
            {
                break;
            }
        }
        if (res)
        {
            for (j = num - 1; j >= i; j--)
            {
                strArray[j + 1] = strArray[j];
            }
            strArray[i] = tmp;
            num++;
            tmp = (char*)malloc((WORD_LENGTH + 1) * sizeof(char));
        }
    }
    free(tmp);
    scanf("}");
    return num;
}
```

杨成熙同学用的是打断字符串的方法，思路非常清楚，同学们可参考他的代码：

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
/*
  该题解思路为，在两个输入的字符串中添加 '\0' 将每个单词都转为字符串，
  并记录指针，如此处理即可做到 “全单词匹配”
*/
```

```
int main()
{
  char in_str_1[10000];        //储存第一个字符串
  char in_str_2[10000];        //储存第二个字符串
  int i,j;
  gets(in_str_1);
  gets(in_str_2);

  char *pt;                    //储存打断后的单词

  char *word_1[101];           //储存第一个字符串中的单词
  int n_1=0;                   //储存第一个字符串中单词个数
  i=0;
  while(1)
  {
    while(!isalpha(in_str_1[i]))
    {
      i++;
      if(in_str_1[i]=='\0')
        break;
    }
    if(in_str_1[i]=='\0')
      break;
    //找到第一个字母，若已经到了字符串尾，则跳出循环
    pt=&in_str_1[i];
    n_1++;
    //记录第一个字母的地址，并将单词数量加一
    while(isalpha(in_str_1[i]))
    {
      i++;
    }
    //找到该单词后面第一个非字母字符
    in_str_1[i]='\0';
    //将其变为 '\0' 实现将该单词变为字符串
    word_1[n_1]=pt;
    //在数组中将该单词的地址记录下来
    //循环结束，i 继续增加，直到找到下一个字母或到达第一个字符串的结尾
  }

  char *word_2[101];
  int n_2=0;
  i=0;
  while(1)
  {
    while(!isalpha(in_str_2[i]))
    {
      i++;
      if(in_str_2[i]=='\0')
        break;
    }
    if(in_str_2[i]=='\0')
      break;
    pt=&in_str_2[i];
    n_2++;
    while(isalpha(in_str_2[i]))
    {
      i++;
    }
    in_str_2[i]='\0';
    word_2[n_2]=pt;
```

```
  }
  //道理同处理第一个字符串

  char destroy[38];
  memset(destroy,' ',36);
  destroy[36]='\0';
  //创建一个用来摧毁单词的字符串，将其变为 36 个空格

  char *word_out[102];       //用来储存需要输出的单词
  int judge,n_out=0;
  //judge 为 0 表示没有匹配到相同单词，为 1 表示匹配到相同单词
  for(i=1;i<=n_2;i++)
  {
    judge=0;
    for(j=1;j<=n_1;j++)
    {
      if(strcmp(word_1[j],word_2[i])==0)
      {
        judge=1;
        strncpy(word_1[j],destroy,strlen(word_2[i]));
        //匹配到单词后，将字符串 1 中的单词摧毁，防止多次匹配
      }
    }
    if(judge==1)
    {
      n_out++;
      word_out[n_out]=word_2[i];
    }
  }

  if(n_out==0)
  {
    printf("NONE\n");
    return 0;
  }

  int isOut[101]={0};            //标记需要输出的单词是否已经输出
  int min_j;                     //记录最小的单词下标
  memset(destroy,'z',37);
  destroy[37]='\0';
  //将摧毁字符串设为 37 个 z
  word_out[101]=destroy;
  //让第 “101”个单词变为摧毁字符串
  for(i=1;i<=n_out;i++)
  {
    min_j=101;
    for(j=1;j<=n_out;j++)
    {
      if(!isOut[j]&&strcmp(word_out[j],word_out[min_j])<=0)
        min_j=j;
    }
    printf("%s ",word_out[min_j]);
    isOut[min_j]=1;
  }
  printf("\n");
  return 0;
}
```

D 傻傻 Aqi 与有趣的 qsort()

时间限制：1000ms　内存限制：65536kb

通过率：329/363 (90.63%)　正确率：329/775 (42.45%)

题目描述

傻傻 Aqi 想用快速排序解决一个排序问题。Alice 表示 C 语言的库函数中本身就包含了一个能实现快速排序算法的函数——qsort。傻傻 Aqi 经过学习与研究，发现它真是一个有趣的函数，这个函数能通过指针移动的方式，根据你给的比较条件进行快速排序。现在来熟悉一下这个函数吧~

输入一个正整数 n，然后输入 n 个整数，对这 n 个整数使用 qsort 函数从小到大排序后，首先输出中间的那个数，若 n 为偶数则输出中间两个数的平均数，然后输出排序后排在数列第 5 个的数。

输入

单组数据输入，

第一行输入一个正整数 n（5<=n<1000),

第二行输入 n 个整数。

输出

对于该组输入，

第一行，若 n 为奇数则输出这 n 个整数排序后中间的那个整数，若 n 为偶数则输出排序后中间两个数的平均数（保留两位小数）；

第二行，输出从小到大排序后排在数列第 5 位的整数（数列从 1 记）。

输入样例

```
12
10 90 -80 -40 20 70 0 100 25 -45 30 50
```

输出样例

```
22.50
10
```

HINT

qsort 函数包含在 stdlib.h 头文件中！

考察知识点

快排的使用
难度系数 3

解题思路

本题考察 qsort 函数的使用。qsort 函数需要 4 个参数，分别为数组名、数组元素个数、数组中元素大小和自定义的比较函数。定义比较函数时需注意函数类型、参数类型及类型转换。另，使用该函数需要包含 stdlib.h 头文件。

代码

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 1000

int cmp(const void* a,const void* b){
   return *(int*)a-*(int*)b;
}

int main(){
   int n;
   scanf("%d",&n);
   if(n<5){
      return 0;
   }
   int num[MAX];
   for(int i=0;i<n;i++){
      scanf("%d",&num[i]);
   }
   qsort(num,n,sizeof(num[0]),cmp);
   if(n%2==0){
      float result=(num[n/2]+num[n/2-1])/2.0;
      printf("%.2f\n",result);
   }else{
      printf("%d\n",num[n/2]);
   }
   printf("%d",num[4]);
   return 0;
}
```

E 更遥远的星期几

时间限制：1000ms 内存限制：65536kb

通过率：845/1040 (81.25%) 正确率：845/3540 (23.87%)

题目描述

大家能轻松的算出下个月的某一天是星期几了，那么时间更长一些呢？
现在已知当前是 X 月 Y 日星期 Z，求现在开始的**下一个 A 月份**第 B 天是周几呢？请用指针数组完成。

输入

五个整数 XYZABXYZAB
XX 月 YY 日星期 ZZ，求 AA 月 BB 日为星期几（Z=7Z=7 表示周日，假定 2 月一直为 28 天）

输出

一行字符串，表示星期几，要求首字母大写。

输入样例

```
4 27 5 5 1
```

输出样例

```
Tuesday
```

题解：
总体思路和之前的一样，统计出两个日期之间的长度再对 7 取模即可，用时间数组来储存每个月的天数，要注意是否会月份到下一年

```
#include<stdio.h>
int m[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
char *w[] =
{ "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" };
int main()
{
	int now,nxt,d_now,w_now,d_nxt;
	scanf("%d%d%d%d%d",&now,&d_now,&w_now,&nxt,&d_nxt);
	//当前月，当前时间，当前星期，目标月，目标时间
	int al = m[now]-d_now;
	now = now%12+1;
	while(now!=nxt)
```

```
	{
		al += m[now];
		now = now%12+1;
	}
	int anw = (al+d_nxt+w_now)%7;
	printf("%s",w[anw]);
}
```

F 解密 ONE

时间限制：1000ms　内存限制：65536kb

通过率：375/607 (61.78%)　正确率：375/1562 (24.01%)

题目描述

上次一位解密员拜托各位帮他找到了文章中最长的一行，但是他发现问题没那么简单，他仍需要其他行的信息。这一次，需要你们将文章每一行按照长度排序后，从最长到最短输出。

输入

多行字符串输入，行数小于 10001000，行长小于 200200，每行只有大小写字母。

输出

按照每行长度，从长到短依次输出每行。如果两行长度相同，则按首字母的字母表顺序从小到大输出，大写字母开头的句子在小写字母之前（保证首字母均不同）。

注意:输出最后一行不能是一个空行,否则可能会导致 PE

输入样例

```
Abcdefg
hijklmn
opqrstuvwxyz
```

输出样例

```
opqrstuvwxyz
Abcdefg
hijklmn
```

考察知识点

```
二维数组，指针
难度系数 4
```

解题思路：

这道题由第六次上机的“阿狄的加密文件”改编而来，最大的变化就是需要将每一句话都存下来，并且按照一定规则排序后输出。先讨论怎么存的问题，一种方法是用二维数组，每一个元素就是一个字符。还有一种方法就是用指针数组，我们让其中的每一个指针各指向一行，但是这里需要用到 malloc()函数，即需要给每一行都分配储存空间，再让指针指向那个空间的地址，这里不多加阐述，感兴趣的可以自己百度了解一下。接下来就是排序的问题，排序可以使用的方法较多，后面给的示例代码中使用的方法是从第一行开始往后找，找到最长的一行后进行交换，再从第二行开始以同样的规则寻找剩下行中的最长行，直到遍历完所有行。

参考代码：
指针数组：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXLENGTH 200
#define MAXLINES 1000
char *lineptr[MAXLINES], buf[MAXLENGTH] ;

int main() {
	int i=0,j,n,max;
	char* pc=buf;
	while(scanf("%s",&buf)!=EOF){
		lineptr[i]=(char*)malloc(strlen(buf)+1);
		strcpy(lineptr[i],buf);
		i++;
	}
	n=i;
	for(i=0;i<n;i++){
		max=i;
		for(j=i+1;j<n;j++){
			if(strlen(lineptr[max])<strlen(lineptr[j]))
				max=j;
			else if(strlen(lineptr[max])==strlen(lineptr[j])){ //如果长度相同
				if(lineptr[max][0]>lineptr[j][0])	//判断首字母先后来排序
					max=j;
			}
		}
		if(max!=i){				//进行交换
			pc=lineptr[max];
			lineptr[max]=lineptr[i];
			lineptr[i]=pc;
		}
	}
	for(i=0;i<n;i++){
		puts(lineptr[i]);
		free(lineptr[i]);
	}
}
```

二维数组：

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#define MAXLENGTH 200
#define MAXLINES 1000
char lines[MAXLINES][MAXLENGTH], buf[MAXLENGTH] ;
/* run this program using the console pauser or add your own getch, system("pause") or input
loop */

int main() {
	int i=0,j,n,max;
	char* pc=buf;
	while(scanf("%s",&buf)!=EOF){
		strcpy(lines[i],buf);
		i++;
	}
	n=i;
	for(i=0;i<n;i++){
		max=i;
		for(j=i+1;j<n;j++){
			if(strlen(lines[max])<strlen(lines[j]))
				max=j;
			else if(strlen(lines[max])==strlen(lines[j])){
				if(lines[max][0]>lines[j][0])
					max=j;
			}
		}
		if(max!=i){
			strcpy(buf,lines[max]);
			strcpy(lines[max],lines[i]);
			strcpy(lines[i],buf);
		}
	}
	for(i=0;i<n;i++){
		puts(lines[i]);
	}
}
```

感兴趣的可以稍微对比下二者的不同。

G PPZ 的字符扩展

时间限制：1000ms 内存限制：65536kb

通过率：395/575 (68.70%) 正确率：395/1363 (28.98%)

题目描述

程序猿一定要学会偷懒！！！简写就是个很好的方式，但是现在 PPZ 需要你来扩展简写后的字符串。

定义：在输入的字符串中，如果有类似“d-h”或者“4-8”的子串，我们就把它当作一种简写，即在输入的字符串中，出现了减号“-”，如果减号右边的字符严格大于左边的字符（ASCII 值），**同时还需要满足两个字符是同类字符(同为小写字母,大写字母或者数字)**,就需要扩展该字符串，用连续递增的字符替代其中的减号，即，将上面两个子串分别输出为“defgh”和“45678”。

如果不满足严格大于条件，则**不能**去掉减号,保持原样输出，如“4-1”输出“4-1”,"3-3"输出"3-3".

输入

输入共一行，为一行输入的简写字符串（保证每个输入字符的 ASCII 值都在[33,126]），保证不会连续出现两个减号，字符串长度不超过 100，保证输入不是空行，保证减号不会出现在字符串的开头和结尾。

输出

输出共一行，为展开后的字符串。

输入样例

```
a-f-g-h-hda-!
```

输出样例

```
abcdefgh-hda-!
```

考察知识点

字符串处理
难度系数 5

解题思路

根据题目叙述判断不存在‘-’在末尾的特殊情况，故先读入字符串然后对其进行遍历，遇到‘-’判断是否满足题目要求的扩展条件，若满足进行扩展，若不满足原样输出，每次输出从左边字母到'-'位置部分，然后再对下一个字符进行判断（即 a-c 类型先输出 ab，再接着判断)。最后注意题目边界条件即可。

例程

```
#include<stdio.h>
          #include<string.h>
          #define N (100+10)
          int judge(char a,char b);
          int main()
          {
              char in[N],t;
              scanf("%s",in);
              int i,len;
              len=strlen(in);
              for(i=0;i<len;++i){
                  if(i==len-1||in[i+1]!='-')
                      printf("%c",in[i]);
                  else{
                      if(in[i]<in[i+2]&&judge(in[i],in[i+2])){
                          for(t=in[i];t<in[i+2];++t)
                              printf("%c",t);
                          i++;
                      }
                      else{
                          printf("%c-",in[i++]);
                      }
                  }
              }
              return 0;
          }
          int is_digit(char a)
          {
              return a>='0'&&a<='9';
          }
          int is_upper(char a)
          {
              return a>='A'&&a<='Z';
          }
          int is_lower(char a)
          {
              return a>='a'&&a<='z';
          }
          int judge(char a,char b)
          {
```

```
    if(is_digit(a)&&is_digit(b))
        return 1;
    else if(is_lower(a)&&is_lower(b))
        return 1;
    else if(is_upper(a)&&is_upper(b))
        return 1;
    return 0;
}
```

H Terry 的大水题

时间限制：1000ms　内存限制：65536kb

通过率：247/481 (51.35%)　正确率：247/1181 (20.91%)

题目描述

发洪水了。

在某个二维空间内有很多岩石排成一条直线，Terry 想知道洪水过后这些岩石间能存多少水呢？

输入

两行，第一行输入一个正整数 nn，代表岩石的个数，0<n≤10000<n≤1000。
第二行输入 nn 个整数，代表这些岩石的高度 hh，h>=0h>=0，岩石的宽度都为 11。

输出

输出一行，表示蓄水的最大值。

输入样例 1

```
3
1 0 1
```

输出样例 1

```
1
```

输入样例 2

```
4
3 1 2 3
```

输出样例 2

```
3
```

HINT

样例 1 代表了一组凹字形的岩石，形如 口_口 ，中间凹下去的部分可以蓄水。

考察知识点

```
数组、循环的综合运用
难度系数 6
```

解题思路

本题给了每块岩石的高度，求总共能蓄水多少体积。我们很容易想到，对于每一块岩石本身，它上方蓄的水的体积等于他左边最高的岩石和右边最高的岩石中较矮的那块与自身的高度差（有点绕）。所以我们先把每块岩石左边最高的高度和右边最高的高度计算出来，然后通过循环将每块石头上方蓄积的水的体积加起来就是答案。注意当只有两块石头的时候是不能蓄水的。具体操作见代码。

标程

```
#include <stdio.h>

long long  a[1001];
long long  l[1001];
long long  r[1001];

long long leftmax(int n)    //计算当前石头左边最高的高度
{
    long long max=0;
    int i;
    for(i=0;i<n;i++)
    {
        if(a[i]>max)
            max=a[i];
    }
    if(max<a[n]) return a[n];
    return max;
}
long long rightmax(int n,int r)  //计算当前石头右边最高的高度
{
    long long max=0;
    int i;
    for(i=n+1;i<=r;i++)
    {
        if(a[i]>max)
            max=a[i];
    }
    if(max<a[n]) return a[n];
    return max;
}
long long less(long long a,long long b)
{
```

```
    if(a>b) return b;
    return a;
}

int main()
{
    int n;
    scanf("%d",&n);
    long long high=0;
    int i;
    for(i=0;i<n;i++)
    {
        scanf("%lld",&a[i]);
    }
    if(n<=2)
    {
        printf("0\n");
    }
    else
    {
        for(i=1;i<n-1;i++)
        {
            l[i]=leftmax(i);
            r[i]=rightmax(i,n-1);
        }
        for(i=1;i<n-1;i++)
        {
            high+=less(l[i],r[i])-a[i];  //计算当前石头上方蓄积的水体积
        }
        printf("%lld\n",high);
    }

}
```

Tarpe 酋长的 Excel

时间限制：1000ms　内存限制：65536kb

通过率：665/815 (81.60%)　正确率：665/2390 (27.82%)

题目描述

给定一个 Excel 表格中显示的列标题，返回其相应的列号

输入

一个字符串（长度不超过 1010）表示列标题。

输出

输出一行，对应的列号。

输入样例 1

```
A
```

输出样例 1

```
1
```

输入样例 2

```
B
```

输出样例 2

```
2
```

输入样例 3

```
AA
```

输出样例 3

```
27
```

考察知识点

```
字符串处理
难度系数 2
```

解题思路：
这道题目考察的其实是 26 进制，如果看出了这点，这道题就非常简单了，for 循环迭代相加即可。

坑点：

8. 输入建议用 scanf 直接读入字符串即可（如果你搞不懂 getchar 和 gets 的终止条件）
9. 抱歉，最后一组数据很大，需要用 long long 输出，这也是需要关注的细节。

AC 示例代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char s[205];

int main()
{

   scanf("%s",&s);
   long long number = 0;
   int len = strlen(s);

   for(int i=0; i<len; i++){
      number = number*26 + s[i] - 'A' + 1;
   }
   printf("%lld\n",number);

}
```

Tarpe 酋长的游戏教学

时间限制：1000ms　内存限制：65536kb

通过率：129/311 (41.48%)　正确率：129/720 (17.92%)

题目描述

很多大型游戏的地图其实只是维度比较高的数组而已。

Tarpe 酋长希望大家能在给定的地图中找到岛的数量。
给定只包含陆地 `1` 和海洋 `0` 的二维数组表示地图，计算岛的数量。一个岛是被水包围，并且通过水平或垂直连接相邻的陆地而形成。
你可以假设这个地图所有四个边都被水包围

输入

第一行两个数字 x,yx,y(0<x,y<200<x,y<20)，表示数组的两个维度的长度，
接下 xx 行，每行一个长度为 yy 的字符串，表示二维数组的元素。

输出

输出一行，岛的数量。

输入样例 1

```
4 5
11110
11010
11000
00000
```

输出样例 1

```
1
```

输入样例 2

```
4 5
11000
11000
00100
00011
```

输出样例 2

```
3
```

考察知识点

```
二维数组的综合应用
难度系数 8
```

解题思路：

这道题考察同学们解决二维数组下复杂问题的能力，首先岛的定义是被水包围，通过水平或者垂直连接相邻的陆地块。那么我们不妨遍历这个数组中的陆地，如果遇到陆地，就判断周围还有没有连接的陆地，有的话就继续前进，没有的话就终止。对于已经访问过的陆地，我们可以对它进行标记，这样下次就不会访问同一块陆地了。如果我们采用递归的方式来描述刚刚的过程，那么只需要计算不重复递归成功的总数，就是岛的总数。

事实上，这个思想就是算法中常用的 dfs，深度优先搜索算法。酋长的代码比较简洁，大家可以借鉴一下。

坑点:

1. 四面环海，注意这个条件
2. 注意二维数组的上下左右四个边界，遍历时不能越界

AC 代码：

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX_N 25

char grid[MAX_N][MAX_N];
int row_n = 0;
int col_n = 0;

int visit(int x, int y)
{
  if(x<0 || y<0 || x>=col_n || y>= row_n || grid[y][x] != '1')
  {
```

```
        return 0;
    }

    grid[y][x] = 'v';

    visit(x+1,y);
    visit(x-1,y);
    visit(x,y+1);
    visit(x,y-1);
    return 1;
}

int main()
{
    scanf("%d%d",&row_n,&col_n);
    for(int i=0;i<row_n;i++)
    {
      scanf("%s",&grid[i]);
    }
    int total = 0;
    for(int y=0; y < row_n;y++)
    {
        for(int x=0; x< col_n; x++)
        {
            total += visit(x,y);
        }
    }
    printf("%d\n",total);
}
```