**程序设计算法练习题**

1. 从键盘上输入后缀表达式，试设计算法计算表达式的值。

（**规定：**逆波兰表达式的长度不超过1行，输入以“$”作为结束，操作数之间用空格分隔，操作符只可能有“+”“-”“\*”“/”4种。**例如：**234 34 + 2\*$）

2. 已知字符串s1中存放一段英文，设计算法format(s1,s2,s3,n)，要求将s1按给定的长度n格式化成两端对齐的字符串存储在s2中（即确保s2的长度为n且首尾字符不得为空格），s1多余的字符存储在字符串s3中。

3. 设二维数组a[1…m, 1…n]含有m\*n个整数，设计一个算法判断a中所有元素是否互不相同，输出相关信息（yes/no），并试分析算法的时间复杂度。

4. 输入一个字符串，输出其中小写字母的个数，字符串长度不超过80。

**【参考代码】**

int main() {

char s[81];

int n;

printf("输入字符串: ");

scanf("%s", &s);

int answer = 0;

for (int i = 0;i < strlen(s);i++) {//判断第i个字符是不是小写字母

if (s[i] >= 'a' && s[i] <= 'z') answer++;

}

printf("小写字母的个数为%d", answer);

return 0;

}

5. 输入一个字符串，把其中所有小写字母都改写成大写字母，字符串长度不超过80。

**【参考代码】**

int main() {

char s[81];

int n;

printf("输入字符串: ");

scanf("%s", &s);

for (int i = 0;i < strlen(s);i++) {//判断第i个字符是不是小写字母

if (s[i] >= 'a' && s[i] <= 'z')

s[i]=s[i]-'a'+'A'; //例如:s[i]='b',b-a+A⇔98-97+65=66(ASCII:'B')

}

printf("转变后的字符串为%s", s);

return 0;

}

6. 给定n个字符串，接下来给你一个待查找字符串，输出待查找字符串出现的序号，字符串长度不超过80，n<=100。（**输入样例：**先输入n，再输入n个字符串，再输出待查找字符串。）

**【参考代码】**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_LEN 85

#define N 105

int main() {

char s[N][MAX\_LEN];

int n;

printf("要输入字符串的个数为");

scanf("%d", &n);

for (int i = 0;i < n;i++) {

printf("输入字符串: ");

scanf("%s", &s[i]); //输入n次长度为MAX\_LEN-1的字符串

}

char ss[MAX\_LEN];

printf("请输入你要查找的字符串为");

scanf("%s", &ss);

for (int j = 0; j < n; j++) {

if (strcmp(ss, s[j]) == 0) {

printf("待查找字符串所在行为%d", j + 1);

break;

}

}

return 0;

}

7. 如果一个字符串可以由某个长度为k的字符串重复多次得到，则称该串以k为周期，**例如：**abcabcabcabc以3为周期（**注意：**它也以6和12为周期）。输入n个长度不超过80的字符串，依次输出它们的最小周期。

**【参考代码】**

#define MAX\_LEN 80

int main() {

int n=0;

char s[MAX\_LEN];

scanf("%d", &n);

while (n--) {

scanf("%s", &s);

int len = strlen(s);

for (int i = 1;i <= len;i++) {

if (len % i == 0) {

int flag = 1;

for (int j = i;j < len;j++) {

if (s[j] != s[j % i]) {

flag = 0;

break;

}

}

if (flag == 1) {

printf("%d\n", i);

break;

}

else continue;

}

}

}

return 0;

}

8. 输入两个字符串s和t，问是否能从t中删除0个或者多个字符（其他字符顺序不变），得到字符串s。**例如：**abcde可以得到bce，但无法得到dc，字符串长度不超过1000。

**样例输入：**每行两个分别为s和t，不包含组数的多组输入。

|  |  |
| --- | --- |
| 1 | sequence subsequence |
| 2 | person compression |
| 3 | VERDI vivaVittorioEmanueleReDiItalia |
| 4 | caseDoesMatter CaseDoesMatter |

**样例输出：**

|  |  |
| --- | --- |
| 1 | Yes |
| 2 | No |
| 3 | Yes |
| 4 | No |

**【参考代码】**

int main() {

char s[1010], t[1010];

while (scanf("%s %s", &s, &t)) {

int j = 0;

for (int i = 0;i < strlen(t);i++) {

if (t[i] == s[j]) j++;

if (j == strlen(s)) {

printf("Yes\n");

break;

}

}

if(j != strlen(s)) printf("No\n");

}

return 0;

}

9. 字符串反转，例如将字符串“www.sddff.”反转为“.ffdds.www”。

**【案例分析】**

**思路：**

（1）定义两个指针（标识），一个指向字符串的开头，一个指向字符串的末尾

（2）交换这两个指针（标识）所指向的字符，并向中间移动指针（标识），**直到两个指针相遇或交叉**。

[**注意：**此处的指针理解成标识更好。]

**【参考代码】**

#include <stdio.h>

#include <stdlib.h>

#include <assert.h>

void reverseString(char\* str) {

int length = strlen(str);

int start = 0;

int end = length - 1;

while (start < end) { //当start=end时，交换停止

char temp = str[start];

str[start] = str[end];

str[end] = temp;

start++;

end--;

}

}

int main() {

char str1[20];

scanf("%s", &str1);

reverseString(str1);

printf("%s", str1);

return 0;

}

10. 回文字符串是指正读反读都一样的字符串，比如“level”和“noon”都是回文字符串，现要求键盘输入一个字符串，然后设计算法判断输入的是否为回文字符串。

**【案例分析】**

**思路：**

可使用双指针（标识）的思想，双指针（标识）分别从字符串的两端向中间移动，比较对应位置的字符是否相等即可。

**【参考代码】**

void PalindromeString(char\* str) {

int length = strlen(str);

int start = 0;

int end = length - 1;

while (start < end) { //当start=end时，判断停止

if (str[start] != str[end]) {

printf("NO\n");

return;

}

else {

start++;

end--;

}

}

printf("YES\n");

}

11. 假设表达式中允许包含3种括号：圆括号、方括号和大括号。设计算法判断给定表达式中的括号是否正确配对。

**【括号匹配中的原则】**

①就近原则

②匹配消除原则

**【思路】**

只要是左括号就入栈（所有种类的左括号），一旦遇到右括号就从堆栈中弹出栈顶元素，与当前右括号去匹配（**注意：**每次弹出栈顶元素前先判断栈是否为空），完成匹配的左右括号就可以删去。

**【参考代码】**

#define true 1

#define false 0

#define bool char

#define MaxSize 100

//栈结构体

typedef struct{

char data[MaxSize];

int top;

int cursize;

}Stack;

//栈初始化

void InitStack(Stack\* S){

S->top = -1;

S->cursize = 0;

}

//栈的判空

bool empty(Stack S) {

if (S.top == -1) return true;

else return false;

}

//入栈

void Push(Stack\* S,char e) {

if (S->cursize == MaxSize) return;

S->data[++S->top] = e;

S->cursize++;

}

//出栈

void Pop(Stack\* S, char e) {

if (S->cursize == 0) return;

int delete = S->data[S->top--];

S->cursize--;

}

int main() {

Stack S;

InitStack(&S);

char str1[50];

scanf("%s", &str1);

int length = strlen(str1);

bool flag = true; //默认为“匹配”

for (int i = 0;i < length;i++) {

if (str1[i] == '(' || str1[i] == '[' || str1[i] == '{')

if(flag) Push(&S, str1[i]);

if (str1[i] == ')' || str1[i] == ']' || str1[i] == '}') {

if (flag && !empty(S)) {

char topchar = S.data[S.top];

if ((topchar == '(' && str1[i] == ')') || (topchar == '[' && str1[i] == ']') || (topchar == '{' && str1[i] == '}')) {

Pop(&S, S.data[S.top]);

}

else {

flag = false;

break;

}

}

}

}

if (flag && empty(S)) printf("YES\n");

else printf("NO\n");

return 0;

}

12. **Switch语句的应用：**要求按照考试成绩的等级输出百分制分数段，A等为85分以上，B等为70~84分，C等为60~69分，D等为60分以下。成绩的等级由键盘输入。

**【题目分析】**

这是一个多路分支选择问题。

**【参考代码】**

int main() {

char grade;

scanf("%c", &grade);

printf("Your score is ");

switch (grade) {

case'A':printf("85~100\n");break;

case'B':printf("70~84\n");break;

case'C':printf("60~69\n");break;

case'D':printf("<60\n");break;

default:printf("enter data is error!\n");

}

return 0;

}

[**注意：**case本身是没有条件判断功能的，程序执行相匹配的case常量后的语句后，无论后面是否还有其他case标号，都会一直执行下去，直到遇到break语句或右花括号“}”为止。]

13. 长度为*n*的环状串有*n*种表示法，分别为从某个位置开始顺时针得到。例如：CTCC有4种表示法：CTCC, TCCC, CCCT, CCTC。我们的任务是输出其中**最小的一个**。对于CTCC应该输出CCCT。*n*<=100。

**样例输入：**

|  |  |
| --- | --- |
| 1 | 2 |
| 2 | CGAGTCAGCT |
| 3 | CTCC |

**样例输出：**

|  |  |
| --- | --- |
| 1 | AGCTCGAGTC |
| 2 | CCCT |

**【参考代码】**

int main() {

char str[200], minstr[200],s[200];

int m;

scanf("%d", &m);

while (m--) {

scanf("%s", &str);

strcpy(minstr, str);

int length = strlen(str);

for (int i = 1;i < length;i++) {

for (int j = 0;j < length;j++) {

s[j] = str[(i + j) % length];

}

s[length] = '\0';

if (strcmp(minstr, s) > 0) {

strcpy(minstr, s);

};

}

printf("%s\n", minstr);

}

return 0;

}

14. 给出一种物质的分子式（不带括号），求分子量。本题中的分子式只包含4种原子，分别为C、H、O、N，原子量分别为12.01，1.008，16.00，14.01。**例如：**C6H5OH的分子量为94.108。字符串长度不超过80。

**样例输入：**

|  |  |
| --- | --- |
| 1 | 4 |
| 2 | C |
| 3 | C6H5OH |
| 4 | NH2CH2COOH |
| 5 | C12H22O11 |

**样例输出：**

|  |  |
| --- | --- |
| 1 | 12.010 |
| 2 | 94.108 |
| 3 | 75.070 |
| 4 | 342.296 |

**【参考代码】**

int main() {

float map[1000]; //新建原子量的映射关系

map['C'] = 12.01;

map['H'] = 1.008;

map['O'] = 16.00;

map['N'] = 14.01;

int n; //分子式个数

scanf("%d", &n);

while (n--) {

int num = 0;

char c=0;

float answer=0;

char s[200];

scanf("%s", &s);

//循环遍历每一个字符

for (int i = 0;i < strlen(s);i++) {

if (s[i] >= 'A' && s[i] <= 'Z') {

if (num == 0 && i != 0) //当i不等于0，说明不是字符串第一个位置，因为数字只能出现在字母(第一位)的后面

num = 1;

answer += map[c] \* num;

c = s[i];

num = 0;

}

else if (s[i] >= '0' && s[i] <= '9') {

num = num \* 10 + s[i] - '0'; //s[i] - '0'代表把s[i]**字符型数字**转成**整形数字**

}

}

if (num == 0)

num = 1;

answer += map[c] \* num;

printf("%.3f\n", answer);

}

return 0;

}

15.