#include <stdio.h>

#include <math.h>

//#include <string.h>

int charAt(const char \*s, char c);

char \*stringcopy(char \*s1, const char \*s2);

char\* stringcat(char\* s1, const char\* s2);

char\* Copy(char\* s1, const char\* s2);

int StrCmp(const char\* s1, const char\* s2);

//16.求字符串的长度。 int length(char \*s);

int length(const char \*s){

int len=0;

for(;s[len]!=0; len++);

return len;

}

//30.将一个字符串s逆置，如"abcd" 逆置后为"dcba"。 char \*reverse(char \*s);

char \*reverse(char \*s){

int len = length(s);

for(int i=0; i<len/2; i++){

char c=s[i]; s[i]=s[len-1-i]; s[len-1-i]=c;

}

return s;

}

//1.将一个整数转换为字符串 void toString(char \*s, int num);

char\* toString(char \*s, int num){

int i=0;

while(num){

s[i++]=num%10+'0';

num /= 10;

}

s[i] = 0;

reverse(s);

return s;

}

//2. 将一个仅含有数字的字符串转换为整数 int str2Int(char \*s);

int str2Int(const char \*s){

int i, num=0;

for(i=0; s[i]!=0; i++){

num = num\*10+(s[i]-'0');

}

return num;

}

//3. 将一个含有数字和小数点的字符串转换为浮点型数据 double str2lf(char \*s);

void split(const char \*s, char \*s1, char \*s2, int pos);

double str2lf(const char \*s)

{

char intPart[10], decPart[20];

int pos = charAt(s, '.');

if(pos == -1){

return str2Int(s);

}

split(s, intPart, decPart, charAt(s, '.'));

double decimal = str2Int(intPart), base=10;

for(int i=1; decPart[i]!=0; i++){

decimal += (decPart[i]-'0')/base;

base \*= 10;

}

return decimal;

}

//4. 求两个正整数的最小公倍数 int lcm(int m, int n);

int lcm(int m, int n){

int k;

for(k=m>n?m:n; k<=m\*n; k++)

if(k%m==0 && k%n==0)break;

return k;

}

//5. 求两个正整数的最大公约数 int gcd(int m, int n);

int gcd(int m, int n){

if(m%n == 0) return n;

return gcd(n, m%n);

}

int gcd2(int m, int n){

int k;

for(k=m<n?m:n; k>0; k--)

if(m%k==0 && n%k==0)break;

return k;

}

//6. 判断一个正整数是否为素数 bool isPrime(int num); 或 int isPrime(int num);

int isPrime(int num){

int i, flag=1, m=sqrt(num);

for(i=2; i<=m; i++)

if(num%i == 0){

flag = 0; break;

}

return flag;

}

//7. 判断一个整数是几位数 int digits(int num);

int digits(int num){

int dig=0;

while(num){

dig++;

num /= 10;

}

return dig;

}

//8. 将一个正整数转换为十六进制表示的字符串 void int2Hex(char \*hex, int num);

void int2Hex(char \*hex, int num){

int i=0;

while(num){

hex[i++] = "0123456789ABCDEF"[num%16];

num /= 16;

}

hex[i] = 0;

reverse(hex);

}

void conv(char \*str, int num, int b){ //转化为b进制数

int i=0;

while(num){

str[i++] = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ"[num%b];

num /= b;

}

str[i] = 0;

reverse(str);

}

//9. 将一个十六进制表示的字符串转换为十进制整数 int hex2Int(char \*hex);

int isNumberChar(char c){

int flag = 0;

if(c >= '0' && c <= '9') flag = 1;

return flag;

}

int hex2Int(char \*hex)

{

int i=0, num = 0;

char c;

while((c=hex[i]) != '\0'){

if(isNumberChar(c)) num = num\*16+(c-'0');

else num = num\*16+(c-'A'+10);

i++;

}

return num;

}

//10. 将一个十进制整数按进制表示法输出（0所在项不输出）

//如：输入 1204 输出 1204=1\*1000+2\*100+4\*1 void int2str(char \*s, int num);

char\* int2str(char \*s, int num){

int dig = 0;

char d[10][20];

int base = 1;

while(num){

//dig++;

int r = num%10;

if(r != 0){

char ss[20];

ss[0] = r+'0'; ss[1] = 0;

stringcat(ss, "\*");

char temp[20];

toString(temp, base);

stringcat(ss, temp);

stringcopy(d[dig++], ss);

}

num /= 10;

base \*= 10;

}

int i;

stringcopy(s, d[dig-1]);

for(i=dig-2; i>=0; i--){

stringcat(s, "+");

stringcat(s, d[i]);

}

return s;

}

//11. 求一个整数所有数字和 int digitsSum(int num);

int digitsSum(int num){

int sum=0;

while(num){

sum += num%10;

num /= 10;

}

return sum;

}

//12. 求一个整数数字逆序后的整数，如输入：1204 输出：4021

int inv(int num){

char s[20];

toString(s, num); //参考题目1.

reverse(s); //参考题目30.

return str2Int(s);//参考题目2.

}

//13. 求两个整数的所有公因子，返回公因子的个数。int factors(int \*fts, int m, int n);

int commonfactors(int \*fts, int m, int n){

int num=0, i, k=m<n?m:n;

for(i=1; i<=k; i++){

if(m%i==0 && n%i==0) fts[num++] = i;

}

return num;

}

//14. 求整数的所有因子和（不含1和它本身）int factorsSum(int m);

int factorsSum(int m){

int i, sum = 0;

for(i=2; i<=m/2; i++){

if(m%i == 0) sum += i;

}

return sum;

}

//15. 输出一个正整数中包含数字k的个数(k为0~9中的任一数字) int numbers(int num, int k);

int numbers(int num, int k){

int total = 0;

while(num){

if(k == num%10) total++;

num/=10;

}

return total;

}

//17. 将字符串s2复制到一个字符串s1中。 char \*stringcopy(char \*s1, const char \*s2);

char \*stringcopy(char \*s1, const char \*s2){

char \*p = s1;

while((\*p++=\*s2++)!=0);

return s1;

}

//18. 将字符串s2连接到字符串s1的后面。 char \*stringcat(char \*s1, const char \*s2);

char \*stringcat(char \*s1, const char \*s2){

char \*p = s1;

while(\*p)p++;

while((\*p++=\*s2++)!=0);

return s1;

}

//19. 将字符串s中的所有大写字母转换为小写字母。 char \*stolwr(char \*s);

char \*stolwr(char \*s){

char \*p = s;

while(\*p){

if(\*p >= 'A' && \*p <= 'Z') \*p += 32;

p++;

}

return s;

}

//20. 将字符串s中的所有小写字母转换为大写字母。 char \*stoupr(char \*s);

char \*stoupr(char \*s){

int i;

for(i=0; s[i]!=0; i++)

if(s[i] >= 'a' && s[i] <= 'z') s[i] -= 32;

/\*char \*p = s;

while(\*p){

if(\*p >= 'a' && \*p <= 'z') \*p -= 32;

p++;

}\*/

return s;

}

//21. 比较两个字符串s1和s2的大小，返回两个字符串中第一个不同字符的ASCII码的差，两个字符串相同，返回0，如："abcd"和"dddd"的结果为-3。 int stringcmp(char \*s1, char \*s2);

int stringcmp(const char \*s1, const char \*s2){

for(;\*s1 && \*s2; s1++, s2++){

if(\*s1-\*s2 != 0) return \*s1-\*s2;

}

if(\*s1==0 || \*s2==0) return \*s1-\*s2;

return 0;

}

//22. 求字符串s中从pos开始，长度为len的子串。 char \*substr(char \*s, char \*subs, int pos, int len);

char \*substr(const char \*s, char \*subs, int pos, int len){

int i=0;

for(i=0; s[i+pos]!=0 && i<len; i++){

subs[i] = s[i+pos];

}

subs[i] = 0;

return subs;

}

//23. 求字符串s中第一次出现字符c的位置，如果不存在返回-1。 int charAt(char \*s, char c);

//int find\_first\_of(char \*s, char c)

int charAt(const char \*s, char c){

int pos = -1, i;

for(i=0; s[i]!=0; i++)

if(s[i] == c){

pos = i; break;

}

return pos;

}

//24. 求字符串s中第一次出现字符串ss的位置， 如果不存在返回-1。 int strAt(char \*s, char \*ss);

//此处使用简单匹配算法，有兴趣的同学请查阅KMP算法

int strAt(char \*s, char \*ss){

int i = 0, pos = -1, lens=length(s), lenss = length(ss);

while(i+lenss < lens){

if(s[i] == ss[0]){

int j;

for(j=1; s[i+j] == ss[j]; j++);

if(j==lenss){

pos = i; break;

}

}

i++;

}

return pos;

}

//25. 求字符串s中元音字母(不区分大小写)的个数。 int vowels(char \*s);

int vowels(char \*s){

int count = 0, i;

for(i=0; s[i]!=0; i++)

switch(s[i]){

case 'a':

case 'A':

case 'e':

case 'E':

case 'i':

case 'I':

case 'o':

case 'O':

case 'u':

case 'U': count++; break;

}

return count;

}

//26. 求字符串s中出现字符c的个数。 int chars(char \*s, char c);

int chars(const char \*s, char c){

int count = 0, i;

for(i=0; s[i]!=0; i++)

if(s[i] == c){

count++;

}

return count;

}

//27. 求一个字符串中包含的所有整数，并返回整数的个数， 如"123ab324xy3cumtb24xyz 17a"中包含5个整数，分别为123, 324, 3, 24, 17。 int integers(char \*s, int nums[]);

int integers(const char \*s, int nums[]){

int total=0, i=0, j;

while(s[i]!=0){

if(isNumberChar(s[i])){

for(j=i+1; s[j]!=0 && isNumberChar(s[j]); j++);

char t[20];

substr(s, t, i, j-i); //求子串 见22.

nums[total] = str2Int(t); // 见2.

i = j+1;

total++;

continue;

}

i++;

}

return total;

}

//27. 求一个字符串中包含的所有整数，并返回整数的个数， 如"123ab324xy3cumtb24xyz 17a"中包含5个整数，分别为123, 324, 3, 24, 17。

int int\_numbers(int nums[], const char\* s){

int i=0, total = 0, isNum = 0, value = 0, j;

char c;

while(c=\*(s+i)){

if(!isNumberChar(c)) {

isNum = 0;

}

else if(isNum == 0){ //连续数字字符组成一个整数

isNum = 1;

value = c - '0';

j = i+1;

while(\*(s+j) && isNumberChar(\*(s+j))){

value = value\*10 + \*(s+j) - '0';

j++;

}

nums[total] = value;

total++;

i = j-1;

}

i++;

}

return total;

}

//28. 将一个字符c插入到字符串s指定的位置上。 char \*insertchar(char \*s, char c, int pos);

char \*insertchar(char \*s, char c, int pos){

int i, len = length(s);

for(i=len; i>pos; i--) s[i] = s[i-1];

s[pos] = c;

s[len+1]=0;

return s;

}

//29. 将一个字符串s2插入到字符串s1指定的位置上。 char \*insertstr(char \*s1, char \*s2, int pos);

char \*insertstr(char \*s1, char \*s2, int pos){

int i, len1 = length(s1), len2 = length(s2);

for(i=len1; i>=pos; i--) s1[i+len2] = s1[i];

for(i=0; i<len2; i++)s1[pos+i]=s2[i];

s1[len1+len2+1]=0;

return s1;

}

//31. 将一个字符串s在指定的位置pos分隔成两个字符串s1和s2。 void split(char \*s, char \*s1, char \*s2, int pos);

void split(const char \*s, char \*s1, char \*s2, int pos){

int i;

for(i=0; i<pos; i++){

s1[i] = s[i];

}

s1[i]=0;

for(i=pos; s[i]!=0; i++){

s2[i-pos] = s[i];

}

s2[i-pos]=0;

}

//32. 求一个字符串s中包含多少个单词（以空格分隔，可能有多个），如"I Love C Language"，包含4个单词。 int words(char \*s);

int words(char \*s){

int count=0, word=0, i;

while(s[i]!=0){

if(s[i] == ' ') word=0;

else if(word == 0){

word = 1; count++;

}

i++;

}

return count;

}

//33. 将一个字符串s中的所有字符old替换为另一个字符new。 char \*replace(char \*s, char old, char new);

char \*replace(char \*s, char old, char new)

{

char \*p = s;

while(\*p){

if(\*p == old) \*p = new;

p++;

}

return s;

}

//34. 二分查找：在一个长度为n的有序数组p中，查找元素x，返回其所在的位置，如果该元素不存在，返回-1。

int binary\_search(int \*p, int n, int x){

int low=0, high=n-1;

while(low <= high){

int mid = (low+high)/2;

if(p[mid] == x) return mid;

if(p[mid] > x) high = mid-1;

else low = mid+1;

}

return -1;

}

//35. 在一个长度为n的数组p中，查找元素x第一次出现的位置，如果该元素不存在，返回-1。

int findx(int \*p, int n, int x){

int i, pos = -1;

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

if(\*(p+i) == x){

pos = i; break;

}

}

return pos;

}

//36. 求字符串s中第一次出现元音字母的位置（不区分大小写），如果不存在返回-1。

int isVowel(char c){ //判断字符是否为原因字母

int flag = 0;

switch(c){

case 'a':

case 'A':

case 'e':

case 'E':

case 'i':

case 'I':

case 'o':

case 'O':

case 'u':

case 'U': flag = 1; break;

}

return flag;

}

int find\_first\_vowel(const char \*s){

int i, pos=-1;

for(i=0; \*(s+i)!=0; i++){

if(isVowel(\*(s+i))) {

pos = i; break;

}

}

return pos;

}

//37. 求字符串s中出现字符c的次数。

//同26.

//38. 在一个长度为n的有序数组p中插入元素x，使得该数组仍然有序。

void insert(int \*p, int n, int x){

int i;

for(i=n-1; i>=0 && \*(p+i) > x; i--){

\*(p+i+1) = \*(p+i);

}

\*(p+i+1) = x;

}

//39. 在字符串s的指定位置上插入另一个字符串。

//同29.

//40 查找n个字符串中，是否存在字符串s，若存在返回第一次出现的位置，否则返回-1.

//若存在返回1，不存在返回0，稍加修改即可

int findstring(char str[][20], int n, const char\* s)

{

int i, pos = -1;

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

if(stringcmp(str[i], s) == 0){

pos = i; break;

}

}

return pos;

}

//41. 用冒泡法对一长度为n的整型数组p排序。

void bubble\_sort(int \*p, int n){

int i, j;

for(i=0; i<n-1; i++){

for(j=0; j<n-1-i; j++){

if(p[j] > p[j+1]){

int t = p[j]; p[j] = p[j+1]; p[j+1] = t;

}

}

}

}

//42. 用选择法对一长度为n的整型数组p排序。

void selection\_sort(int \*p, int n){

int i, j;

for(i=0; i<n-1; i++){

int k = i;

for(j=i+1; j<n; j++){

if(p[j] < p[k])k=j;

}

if(k!=i){

int t=p[i]; p[i]=p[k]; p[k]=t;

}

}

}

//43. 使用插入排序法对一长度为n的整型数组a排序

void insertion\_sort(int \*p, int n){

int i, j;

for(i=1; i<n; i++)

insert(p, i, p[i]); //38. 向有序数组中插入

}

//44. 将长度分别为m和n的有序整型数组a和b，合并到整型数组c中，使得c仍然有序。

void merge(int \*dest, int \*p, int m, int \*q, int n){

int i=0, j=0, k=0;

while(i<m && j<n){

if(p[i] < q[j]) dest[k++] = p[i++];

else dest[k++] = q[j++];

}

if(i==m){

while(j<n) dest[k++] = q[j++];

}

if(j == n){

while(i<m) dest[k++] = p[i++];

}

}

//45 用冒泡法、选择法或插入排序法对n个字符串排序。

void bubble\_sort\_strings(char s[][20], int n){

int i, j;

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

for(j=0; j<n-1-i; j++){

if(StrCmp(s[j], s[j+1])>0){

char t[80];

Copy(t, s[j]);

Copy(s[j], s[j+1]);

Copy(s[j+1], t);

}

}

}

}

//46 用冒泡法、选择法或插入排序法对n个字符串排序。

void selection\_sort\_strings(char s[][20], int n){

int i, j;

for(i=0; i<n-1; i++){

int k=i;

for(j=i+1; j<n; j++){

if(StrCmp(s[j], s[k])<0) k = j;

}

if(k!=i){

char t[80];

Copy(t, s[i]);

Copy(s[i], s[k]);

Copy(s[k], t);

}

}

}

//47 用冒泡法、选择法或插入排序法对n个字符串排序。

void insert\_sort\_string(char s[][20], int n, char \*ss) {

int i;

for(i=n-1; i>=0 && StrCmp(s[i], ss)>0; i--){

Copy(s[i+1], s[i]);

} ;

Copy(s[i+1], ss);

}

void insertion\_sort\_strings(char s[][20], int n){

int i, j;

for(i=1; i<n; i++)

insert\_sort\_string(s, i, s[i]);

}

//48. 统计一个字符串中出现不同字符的个数，如字符串"ababc3x"中含有5个不同字符。

int characters(const char \*s){

int i, k=0, total = 0, m = length(s);

char ss[m]; //用于存储s中所有不同的字符，最多m个

for(i=0; i<m; i++){

if(-1 == charAt(ss, s[i])) ss[k++] = s[i];

}

return k;

}

//49. 写一个函数判断一个年份是否为闰年

int isLeapYear(int year){

int flag = 0;

if(year%4==0 && year%100!=0 || year%400==0)flag = 1;

return flag;

}

//50. 统计两个年份之间闰年的个数

int totalLeapYears(int from, int end){

int i, count=0;

for(i=from; i<=end; i++)

if(isLeapYear(i)) count++;

return count;

}

//模式匹配，BF暴力算法

int bf(const char\*s, const char\* t){

int i=0, j=0, k=0, tlen = length(t);

for(k=0; s[k+tlen]!=0; k++){

i=k;

for(j=0;j<tlen; j++){

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

}

if(j == tlen) break;

}

if(k+tlen <= length(s))

return k;

return -1;

}

//字符串连接

char\* StrCat(char\* s1, const char\* s2){

int i, j;

for(i=0; \*(s1+i) != 0; i++);

for(j=0; \*(s2+j)!=0; j++) \*(s1+i++) = \*(s2+j);

\*(s1+i) = 0;

return s1;

}

//字符串比较

int StrCmp(const char\* s1, const char\* s2){

int i;

for(i=0; \*(s1+i) == \*(s2+i) && (\*(s1+i) != 0 || \*(s2+i) != 0);i++);

return \*(s1+i) - \*(s2+i);

}

//字符串复制

char\* Copy(char\* s1, const char\* s2){

int i;

for(i=0; \*(s2+i)!=0; i++) \*(s1+i) = \*(s2+i);

\*(s1+i) = 0;

return s1;

}

//转大写

char\* StrUpr(char\* s){

int i;

for(i=0; \*(s+i); i++){

if(\*(s+i) >= 'a' && \*(s+i) <= 'z') \*(s+i) -= 32;

}

return s;

}

//转小写

char\* StrLwr(char \*s){

int i;

for(i=0; \*(s+i); i++){

if(\*(s+i) >= 'A' && \*(s+i) <= 'Z') \*(s+i) += 32;

}

return s;

}

//查找字符首次出现的位置

int charAt2(const char\* s, char c){

int i, pos = -1;

for(i=0; \*(s+i); i++){

if(\*(s+i) == c) {pos = i;break;

}

}

return pos;

}

//判断字符是否为数字字符

int isNumberChar2(char c){

if(c >= '0' && c <= '9') return 1;

return 0;

}

//统计字符串中整数个数

int numbers2(int nums[], const char\* s){

int i=0, total = 0, isNum = 0, value = 0, j;

char c;

while(c=\*(s+i)){

if(!isNumberChar(c)) {

isNum = 0;

}

else if(isNum == 0){

isNum = 1;

value = c - '0';

j = i+1;

while(\*(s+j) && isNumberChar(\*(s+j))){

value = value\*10 + \*(s+j) - '0';

j++;

}

nums[total] = value;

total++;

i = j-1;

}

i++;

}

return total;

}