法律声明

- 本课件包括：演示文稿，示例，代码，题库，视频和声音等，小象学院拥有完全知识产权的权利；只限于善意学习者在本课程使用，不得在课程范围外向任何第三方散播。任何其他人或机构不得盗版、复制、仿造其中的创意，我们将保留一切通过法律手段追究违反者的权利。

关注 小象学院

第四课　　字符串

林沐

内容概述

1.字符串的引入
2.字符串的声明与初始化
3.字符串的结束符
4.字符串的输入输出
5.例1-整数的提取
6.字符串数组
7.字符串与函数
8.字符串操作1:求字符串长度
9.字符串操作2:复制字符串
10.字符串操作3:连接字符串
11.字符串操作4:比较字符串
12.字符串操作5:字符串子串查找
13.没有结束符的字符串
14.例2-字符串拆分与排序
15.例3-二进制加法
16.数组统计字符串中字符数量
17.例4:最长回文串

字符串的引入

编写程序的大多情况下，我们需要将**文本**看作一个**整体进行处理**，这时就需要使用**字符串**，C语言中使用**char类型的数组**元素存储与处理字符串。下面这个例子展示了**字符串的输入、输出与遍历**。

```
#include <stdio.h>

int main(){
    char name[100];
    char something[100];                    //字符串(字符数组)声明

    printf("What is your name?\n");
    printf("Please input: ");               //屏幕打印字符串常量

    scanf("%s", name); //读入一个字符串至name，直到遇到换行、空格等。
    getchar();  //从标准输入接收一个换行符

    printf("Say something to us:\n");

    gets(something); //从标准输入读入一个字符串至something，
                                            直到遇到换行字符
    printf("Your name is \"%s\".\n", name);
    printf("You say \"%s\".\n", something);
    printf("Your name is spelled:\n");

    int i = 0;
    while(name[i]){
        printf("%c\n", name[i]);            //遍历字符串一个字符一个字符的输出
        i++;
    }

    return 0;
}
```

```
What is your name?
Please input: LinMu
Say something to us:
I love coding.
Your name is "LinMu".
You say "I love coding.".
Your name is spelled:
L
i
n
M
u
请按任意键继续. . .
```

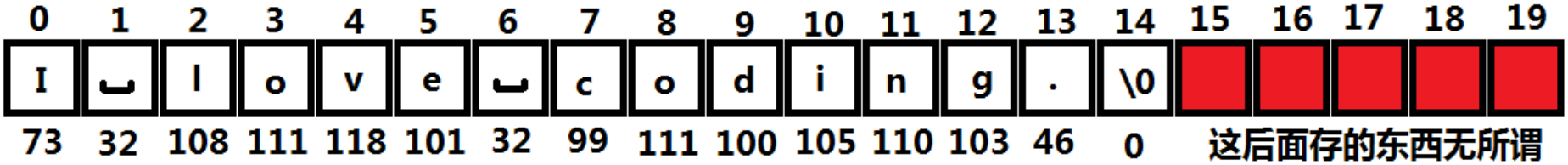
字符串的声明与初始化

字符串由一系列非‘\0’字符(数值为0)与**字符串结束符**‘\0’组成，C语言中使用**字符数组**存储字符串。字符串**不是**C语言的**基本数据类型**，所以无法通过**运算符**实现字符串基本运算；但C语言提供了强大的**字符串处理函数**来实现对字符串的各种操作功能。**初始化后**，编译器在字符串的结尾自动添加'\0'字符，来标识字符串的**结束**。

例如，声明一个字符数组，并将一个字符串初始化到该数组中:

```
char str1[20] = "I love coding.";
```

初始化时，编译器自动在字符串的结尾添加'\0'

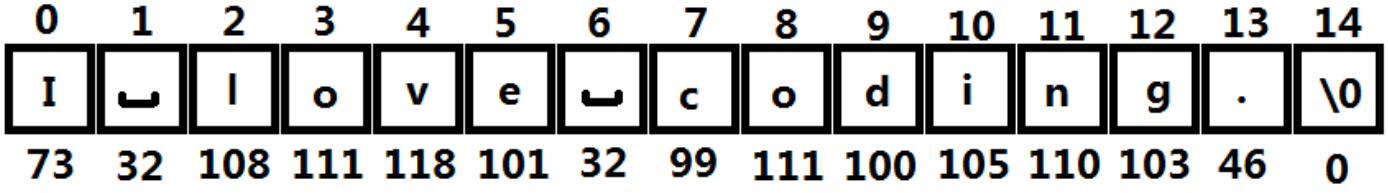


这后面存的东西无所谓

因为对于字符串，\0

就是结尾

```
char str2[] = "I love coding.";
```



字符串的声明与初始化，举例

```c
#include <stdio.h>
int main(){
    char str1[20] = "I love coding.";
    printf("sizeof(str1) = %d\n", sizeof(str1));
    printf("str1 = [%s]\n", str1);
    int i = 0;
    while(str1[i]){
        printf("[%c] : %d\n", str1[i], str1[i]);
        i++;
    }
    printf("\n");
    char str2[] = "I love coding.";
    printf("sizeof(str2) = %d\n", sizeof(str2));
    printf("str2 = [%s]\n", str2);
    i = 0;
    while(str2[i]){
        printf("[%c] : %d\n", str2[i], str2[i]);
        i++;
    }
    return 0;
}
```

//声明一个20个元素的字符数组，存储"I love coding."字符串

//打印字符数组str1的长度

//遇到\0时，停止循环

//一个字符一个字符的遍历并打印

//声明一个字符数组，其长度根据字符串来决定

//打印字符数组str2的长度

```
sizeof(str1) = 20
str1 = [I love coding.]
[I] : 73
[ ] : 32
[l] : 108
[o] : 111
[v] : 118
[e] : 101
[ ] : 32
[c] : 99
[o] : 111
[d] : 100
[i] : 105
[n] : 110
[g] : 103
[.] : 46

sizeof(str2) = 15
str2 = [I love coding.]
[I] : 73
[ ] : 32
[l] : 108
[o] : 111
[v] : 118
[e] : 101
[ ] : 32
[c] : 99
[o] : 111
[d] : 100
[i] : 105
[n] : 110
[g] : 103
[.] : 46
请按任意键继续. . .
```

字符串的结束符

‘\0’字符的**数值**为0，它标识一个字符串的**结束**，即使字符数组中存储了更多的字符，在字符\0后的字符对于该字符串本身**没有太多意义**。对于字符串的各种计算(如求长度、复制、连接等)，\0后的字符**不参与运算**的。

```c
#include <stdio.h>
int main(){
    char ch = '\0';
    printf("ch = %d\n", ch);                    //定义一个\0字符并打印

    char str[20] = "I love\0coding.";           //定义一个字符串存储至str字符数组，在字符e后添加了\0字符
    printf("str = [%s]\n", str);

    int i = 0;
    while(str[i]){
        printf("[%c] : %d\n", str[i], str[i]);
        i++;                                    //一个一个遍历字符，直到遇到\0字符
    }
    printf("\n");

    for (i = 0; i < 20; i++){                   //打印字符数组的全部字符
        printf("i = %d [%c][%d]\n", i, str[i], str[i]);
    }
    printf("\n");

    str[3] = '\0';                              //将字符数组的第4个位置，赋为\0，并打印
    printf("str = [%s]\n", str);

    printf("[%s]\n", &str[10]);                 //讲到指针就真正理解这句话了

    return 0;
}
```

```
ch = 0
str = [I love]
[I] : 73
[ ] : 32
[l] : 108
[o] : 111
[v] : 118
[e] : 101

i = 0 [I][73]
i = 1 [ ][32]
i = 2 [l][108]
i = 3 [o][111]
i = 4 [v][118]
i = 5 [e][101]
i = 6 [ ][0]
i = 7 [c][99]
i = 8 [o][111]
i = 9 [d][100]
i = 10 [i][105]
i = 11 [n][110]
i = 12 [g][103]
i = 13 [.][46]
i = 14 [ ][0]
i = 15 [ ][0]
i = 16 [ ][0]
i = 17 [ ][0]
i = 18 [ ][0]
i = 19 [ ][0]

str = [I ]
[ing.]
请按任意键继续. . .
```

字符串的输入与输出

C语言提供了丰富的字符串输入输出函数，最长用的5个标准输入输出函数: printf、scanf、puts、gets、getchar。

标准输入流stdin:

scanf("%s", name); 将标准输入流中的LinMu读入字符数组name中 遇到空格、换行符、tab符等停止读入

char ch = getchar(); 语句将\n字符读入ch中， 若不加该语句，gets(something);就无法正确读取后续内容。

gets(something); 读取输入流中的字符，直到遇到换行符\n

```c
#include <stdio.h>
int main(){
    char name[100];
    char something[100];
    puts("What is your name?");
    printf("Please input: ");
    scanf("%s", name);
    char ch = getchar();
    printf("Say something to us:\n");
    gets(something);
    printf("Your name is \"%s\".\n", name);
    printf("You say \"%s\".\n", something);
    printf("ch = [%c][%d]\n", ch, ch);
    return 0;
}
```

```
What is your name?
Please input: LinMu
Say something to us:
I love coding.
Your name is "LinMu".
You say "I love coding.".
ch = [
][10]
请按任意键继续. . .
```

例1-整数的提取

问题描述：键盘读入一个**字符串**，其中包括除了空格、换行符、tab符等空字符以外的任意字符，将其中的**整数拆解出**并打印。

输入与输出要求：输入一个字符串(长度<100)，输出字符串中的整数(int范围内)，每个整数占一行。不必考虑负号"-"的问题，即输出的都**是非负数**。

Input Sample:

asdfasoidfj12312@isdjf023#0041#00**999**

Output Sample:

12312

23

41

0

999

Input Sample:

asd--512312@isdjf-7#01#2#345#**

Output Sample:

512312

7

0

1

2

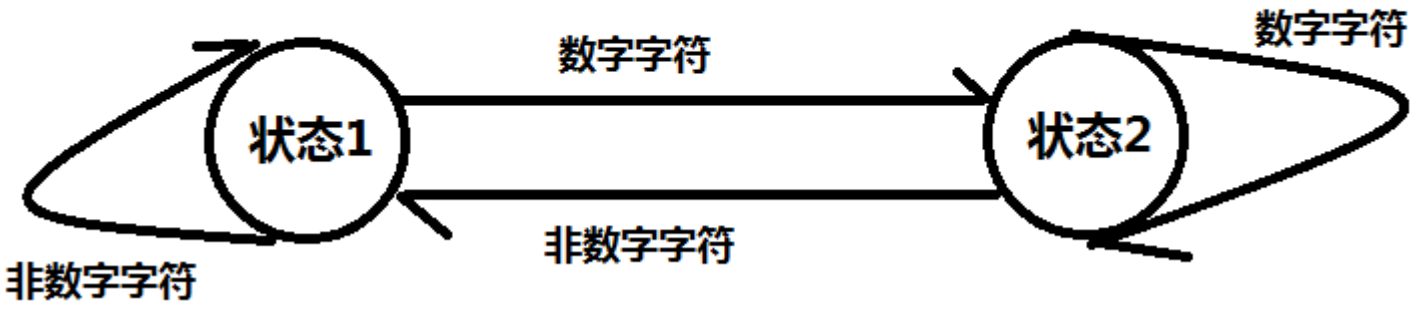
345

用1分钟**思考**解决该问题的算法，字符串如何读入、遍历、处理等。

例1-算法设计

设置字符数组str[100]，存储键盘输入的字符串；
设置变量number = -1，存储提取出来的整数，并且:
　　当number==-1时，代表状态1，正在处理非数字字符。
　　当number!=-1时，代表状态2，正在处理数字字符。
使用变量i遍历字符串，直到遇到'\0':
　　若number==-1时(状态1):
　　　　如果str[i]为数字字符，将该字符转为整型，赋值给number。
　　否则(number!=-1)(状态2):
　　　　如果str[i]为数字字符，将该数字merge进入number。
　　　　否则，打印该数字，重新赋值number = -1。
如果number!=-1: 打印number。

asdfasoidfj12312@isdjf023#0041#**00**999



例1-课堂练习

```
#include <stdio.h>
int main(){
    char str[100];
    int number = -1;
    scanf("%s", str);
    int i;
    for (i = 0; 1 ; i++){
        if (number == -1){
            if ( 2 ){
                number = str[i] - '0';
            }
        }
        else{
            if (str[i] >= '0' && str[i] <= '9'){
                3
            }
            else{
                printf("%d\n", number);
                4
            }
        }
    }
    if ( 5 ){
        printf("%d\n", number);
    }
    return 0;
}
```

3分钟，填写代码，有问题提出！

例1-实现与测试

```
#include <stdio.h>
int main(){
    char str[100];
    int number = -1;
    scanf("%s", str);
    int i;
    for (i = 0; str[i] ; i++){
        if (number == -1){
            if ( str[i] >= '0' && str[i] <= '9' ){
                number = str[i] - '0';
            }
        }
        else{
            if (str[i] >= '0' && str[i] <= '9'){
                number = number * 10 + str[i] - '0';
            }
            else{
                printf("%d\n", number);
                number = -1;
            }
        }
    }
    if ( number != -1 ){
        printf("%d\n", number);
    }
    return 0;
}
```

```
asdfasoidfj12312@isdjf023#0041#**00**999
12312
23
41
0
999
请按任意键继续. . .
```

字符串数组

使用char类型的**二维数组**存储多个字符串，实现**字符串数组**。数组的**每一行**都用来存储一个字符串。

字符串数组的**初始化**可以**省略**二维数组第一维的长度，但必须指定第二维的长度。

在读取或者使用时，使用字符串数组的**第一维来索引**各个字符串。

```
str_array1:
I love coding.
Let's go to work.
Programming is not so hard.
Please input 3 strings:
aaa
bbbbb
ccccccc
str_array1:
aaa
bbbbb
ccccccc
str_array2_size = 3
str_array2:
I love coding.
Let's go to work.
Programming is not so hard.
请按任意键继续. . .
```

```c
#include <stdio.h>
int main(){
    char str_array1[3][50] = {
        "I love coding.",
        "Let's go to work.",
        "Programming is not so hard."
    };
    int i;
    printf("str_array1:\n");
    for (i = 0; i < 3; i++){
        printf("%s\n", str_array1[i]);
    }
    printf("Please input 3 strings:\n");
    for (i = 0; i < 3; i++){
        scanf("%s", str_array1[i]);
    }
    printf("str_array1:\n");
    for (i = 0; i < 3; i++){
        printf("%s\n", str_array1[i]);
    }
    char str_array2[][50] = {
        "I love coding.",
        "Let's go to work.",
        "Programming is not so hard."
    };
    int str_array2_size = sizeof(str_array2) / sizeof(str_array2[0]);
    printf("str_array2_size = %d\n", str_array2_size);
    printf("str_array2:\n");
    for (i = 0; i < 3; i++){
        printf("%s\n", str_array2[i]);
    }
    return 0;
}
```

//定义一个3*50的二维字符数组，存储3个字符串

//打印这三个字符串

//从键盘读取三个字符串保存至字符串数组中

//不指定第一维长度，声明保存三个字符串的二维字符数组

//计算str_array2第一维的长度

字符串与函数

使用**字符串**作为函数的参数，**等同于**使用一维字符数组作为函数的参数，由于字符串以'\0'结尾，故**无需额外的变量**指定使用的字符数组长度；使用**字符串数组**作为函数的参数，**等同于**使用二维字符数组作为函数的参数，需要有一个变量指定字符串数组中有**多少个有效字符串**。通常使用**字符指针**作为参数的写法代替一维数组作为参数的写法，来标明字符串作为函数的参数。

```
#include <stdio.h>
//等同于 int count_str_letters(char str[])
int count_str_letters(char *str){
    int count = 0;
    int i = 0;      //统计str指向的字符串中字母的个数
    while(str[i]){
        if (str[i] >= 'a' && str[i] <= 'z' ||
            str[i] >= 'A' && str[i] <= 'Z'){
            count++;
        }
        i++;
    }
    return count;
}
//统计字符串数组中字母个数
int count_str_array_letters(char str[][100], int n){
    int count = 0;
    int i;
    for (i = 0; i < n; i++){
        count += count_str_letters(str[i]);
    }
    return count;
}
```

```
int main(){
    char str_array[20][100];
    int n;
    printf("Please input strings number:\n");
    scanf("%d", &n);
    int i;
    printf("Please input strings:\n");
    for (i = 0; i < n; i++){
        scanf("%s", str_array[i]);
    }
    printf("letters = %d\n",
        count_str_array_letters(str_array, n));
    return 0;
}
```

```
Please input strings number:
3
Please input strings:
###$!#$abc#abc#abc
abcdefg
99999
letters = 16
请按任意键继续. . .
```

字符串操作1:求字符串长度

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

int my_strlen(const char *str){
    int i = 0;
    while(str[i]){          //普通的求字符串长度的代码
        i++;
    }
    return i;
}

int my_strlen_s(const char *str, int size){
    int i = 0;
    while(str[i] && i < size ){
        i++;
    }                       //安全的求字符串长度代码
    return i;
}

int main(){
    char str[10] = "123456789";
    printf("str = [%s]\n", str);
    printf("strlen = %d\n", strlen(str));
    printf("my_strlen = %d\n", my_strlen(str));
    printf("my_strlen_s = %d\n", my_strlen_s(str, sizeof(str)));

    char str2[5] = "123456789";
    printf("str2 = [%s]\n", str2);
    printf("strlen = %d\n", strlen(str2));
    printf("my_strlen = %d\n", my_strlen(str2));
    printf("my_strlen_s = %d\n", my_strlen_s(str2, sizeof(str2)));
    return 0;
}
```

C语言**标准库**里提供了大量的字符串操作函数，字符串相关的函数全部被包含在头文件<string.h>中。**求字符串长度**这一功能在字符串操作中使用的非常频繁。以下为调用C语言标准库(strlen)求长度、**安全的**求字符串长度的实现代码。

```
str = [123456789]
strlen = 9
my_strlen = 9
my_strlen_s = 9
str2 = [12345(
;?    b◄xu123456789]
strlen = 25
my_strlen = 25
my_strlen_s = 5
请按任意键继续. . .
```

字符串操作2:复制字符串

字符串数据类型是**字符数组**实现的，**没有赋值操作**，故需要使用**字符串复制函数**，将一个字符数组中的内容**复制**到另一个字符数组中，从而实现对**字符串的赋值**。

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

void my_strcpy(char *to, const char *from){
    int i = 0;
    while(from[i]){
        to[i] = from[i];
        i++;
    }
    to[i] = '\0';
}

void my_strcpy_s(char *to, int size, const char *from){
    int i = 0;
    size--;
    while(from[i] && i < size){
        to[i] = from[i];
        i++;
    }
    to[i] = '\0';
}
```

//将from指向的字符串(字符数组)的内容拷贝到to指向的字符数组中

//将from指向的字符串(字符数组)的内容拷贝到to指向的字符数组中，但最多只能拷贝size-1个字符，size是字符数组to的大小

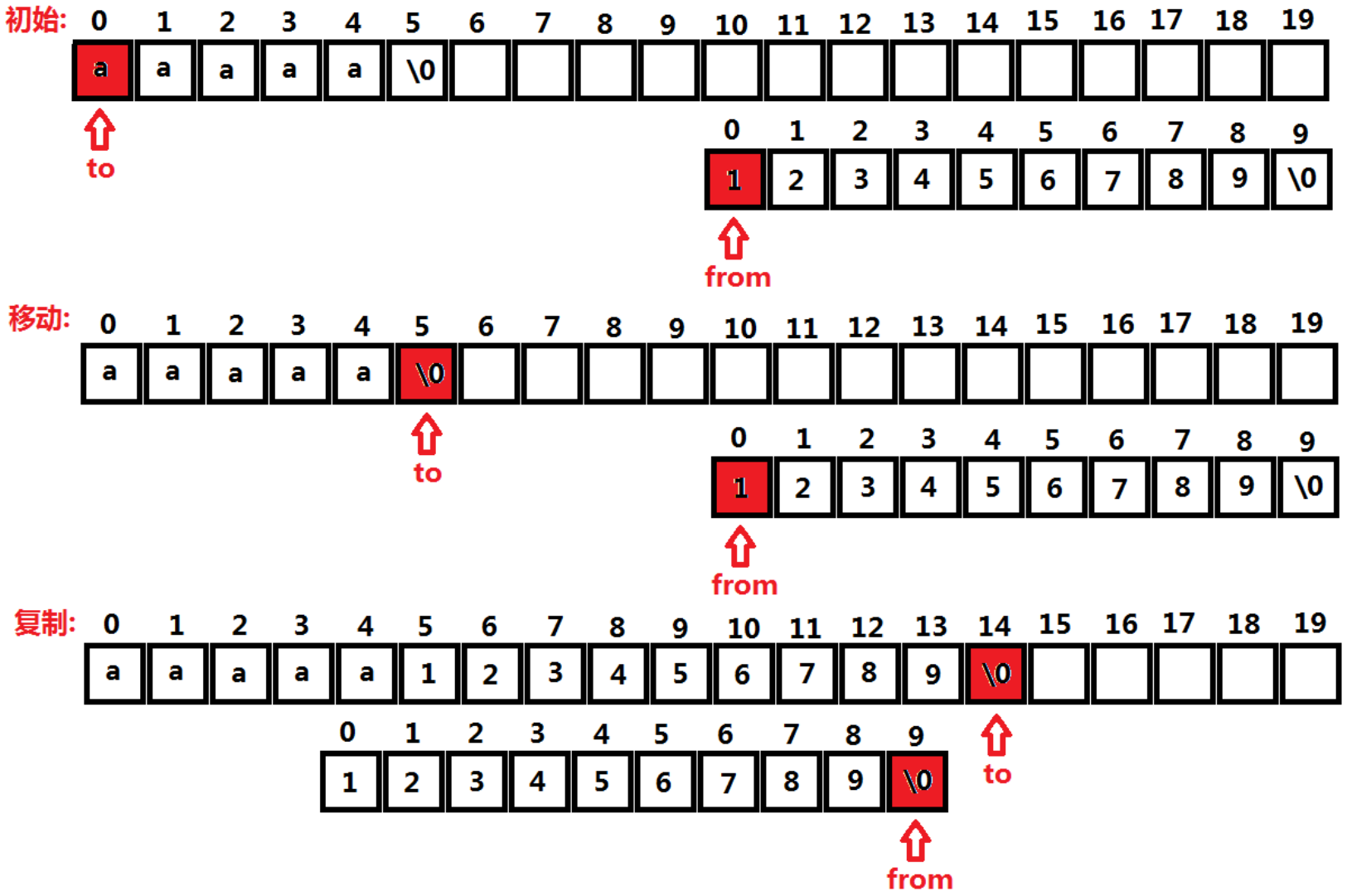
```
int main(){
    char str[10] = "123456789";
    char str2_1[20] = "123456789123456789";
    char str2_2[20] = "123456789123456789";
    char str2_3[20] = "123456789123456789";
    printf("str = [%s]\n", str);
    printf("str2_1 = [%s]\n", str2_1);
    printf("str2_2 = [%s]\n", str2_2);
    printf("str2_3 = [%s]\n", str2_3);
    strcpy(str2_1, str);
    my_strcpy(str2_2, str);
    my_strcpy_s(str2_3, 20, str);
    printf("str2_1 = [%s]\n", str2_1);
    printf("str2_2 = [%s]\n", str2_2);
    printf("str2_3 = [%s]\n", str2_3);

    char str3[5];
    my_strcpy_s(str3, sizeof(str3), str);
    printf("str3 = [%s]\n", str3);
    return 0;
}
```

```
str = [123456789]
str2_1 = [123456789123456789]
str2_2 = [123456789123456789]
str2_3 = [123456789123456789]
str2_1 = [123456789]
str2_2 = [123456789]
str2_3 = [123456789]
str3 = [1234]
```

字符串操作3:连接字符串

将一个字符串的内容**连接到**另外一个字符串内容的**尾部**。



字符串操作3:连接字符串，课堂练习

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

void my_strcat(char *to, const char *from){
    int i = 0;
    while(to[i]){
        i++;
    }
    int j = 0;
    while(from[j]){
        2
        i++;
        j++;
    }
    3
}
```

//将from指向的字符串的内容连接到to指向的字符串内容的尾部。

```
void my_strcat_s(char *to, int size, const char *from){
    int i = 0;
    while(to[i]){
        i++;
    }
    size--;
    int j = 0;
    while(1){
        2
        i++;
        j++;
    }
    3
}
```

//将from指向的字符串的内容连接到to指向的字符串内容的尾部，size变量标识to字符数组的空间，保证安全。

3分钟，填写代码，有问题提出！

```
int main(){
    char str[10] = "123456789";
    char str2_1[20] = "aaaaa";
    char str2_2[20] = "aaaaa";
    char str2_3[20] = "aaaaa";
    printf("str = [%s]\n", str);
    printf("str2_1 = [%s]\n", str2_1);
    printf("str2_2 = [%s]\n", str2_2);
    printf("str2_3 = [%s]\n", str2_3);
    strcat(str2_1, str);
    my_strcat(str2_2, str);
    my_strcat_s(str2_3, 20, str);
    printf("str2_1 = [%s]\n", str2_1);
    printf("str2_2 = [%s]\n", str2_2);
    printf("str2_3 = [%s]\n", str2_3);
    char str3[10] = "aaaaa";
    my_strcat_s(str3, sizeof(str3), str);
    printf("str3 = [%s]\n", str3);
    return 0;
}
```

```
str = [123456789]
str2_1 = [aaaaa]
str2_2 = [aaaaa]
str2_3 = [aaaaa]
str2_1 = [aaaaa123456789]
str2_2 = [aaaaa123456789]
str2_3 = [aaaaa123456789]
str3 = [aaaaa1234]
请按任意键继续. . .
```

字符串操作3:连接字符串，实现

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

void my_strcat(char *to, const char *from){
    int i = 0;
    while(to[i]){
        i++;
    }
    int j = 0;
    while(from[j]){
        to[i] = from[j];
        i++;
        j++;
    }
    to[i] = '\0';
}
```

//将from指向的字符串的内容连接到to指向的字符串内容的尾部。

```c
void my_strcat_s(char *to, int size, const char *from){
    int i = 0;
    while(to[i]){
        i++;
    }
    size--;
    int j = 0;
    while(from[j] && i < size){
        to[i] = from[j];
        i++;
        j++;
    }
    to[i] = '\0';
}
```

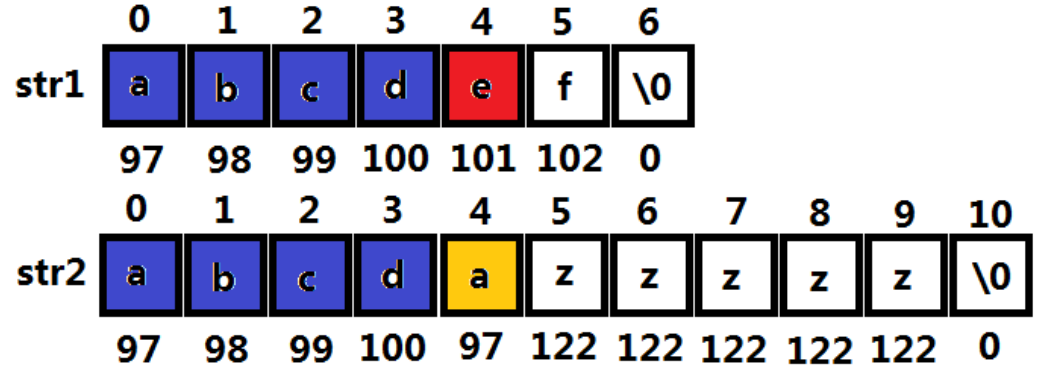
//将from指向的字符串的内容连接到to指向的字符串内容的尾部，size变量标识to字符数组的空间，保证安全。

```c
int main(){
    char str[10] = "123456789";
    char str2_1[20] = "aaaaa";
    char str2_2[20] = "aaaaa";
    char str2_3[20] = "aaaaa";
    printf("str = [%s]\n", str);
    printf("str2_1 = [%s]\n", str2_1);
    printf("str2_2 = [%s]\n", str2_2);
    printf("str2_3 = [%s]\n", str2_3);
    strcat(str2_1, str);
    my_strcat(str2_2, str);
    my_strcat_s(str2_3, 20, str);
    printf("str2_1 = [%s]\n", str2_1);
    printf("str2_2 = [%s]\n", str2_2);
    printf("str2_3 = [%s]\n", str2_3);
    char str3[10] = "aaaaa";
    my_strcat_s(str3, sizeof(str3), str);
    printf("str3 = [%s]\n", str3);
    return 0;
}
```

```
str = [123456789]
str2_1 = [aaaaa]
str2_2 = [aaaaa]
str2_3 = [aaaaa]
str2_1 = [aaaaa123456789]
str2_2 = [aaaaa123456789]
str2_3 = [aaaaa123456789]
str3 = [aaaaa1234]
请按任意键继续. . .
```

字符串操作4:比较字符串

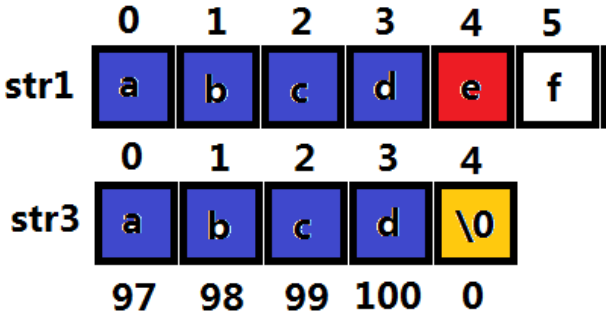
字符串是可以**比较**的，即可以确定一个字符串**是大于还是小于**另一个字符串。两个字符串的比较基于它们中的字符的**ASC2码**，比较两个字符串**第一个不相等**的字符的ASC2码，谁的字符大，哪个字符串就大。



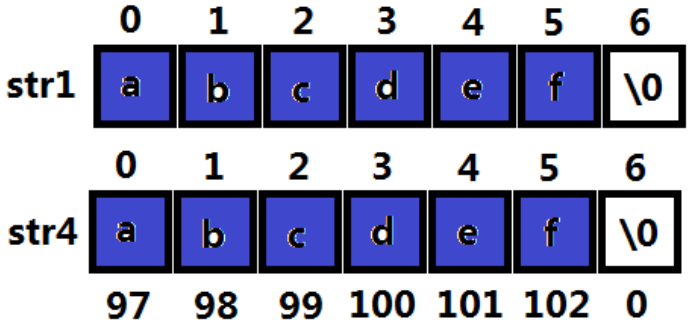
第一个不相等的字符 str1[4] > str2[4]

所以，strcmp(str1, str2) > 0 (大于时返回1)

即字符串str1大于字符串str2

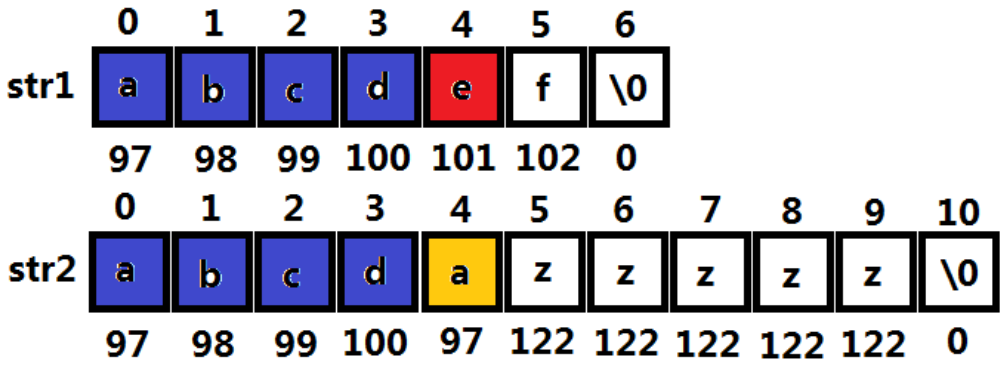


strcmp(str1, str3) > 0



strcmp(str1, str4) == 0

字符串操作4:比较字符串，课堂练习



第一个不相等的字符 str1[4] > str2[4]

所以，strcmp(str1, str2) > 0 (大于时返回1)

即字符串str1大于字符串str2

```
int my_strcmp(const char *str1, const char *str2){
    int i = 0;
    while(str1[i] && str2[i] &&  1  ){
        i++;
    }
    if (  2  ){
        return 1;
    }
    else if(str1[i] < str2[i]){
        return -1;
    }
     3 
}
```

3分钟，填写代码，有问题提出！

字符串操作4:比较字符串，实现

```
int my_strcmp(const char *str1, const char *str2){
    int i = 0;
    while(str1[i] && str2[i] && str1[i] == str2[i]){
        i++;
    }
    if (str1[i] > str2[i]){
        return 1;
    }
    else if(str1[i] < str2[i]){
        return -1;
    }
    return 0;
}
int main(){
    char str1[] = "abcdef";
    char str2[] = "abcdazzzzz";
    char str3[] = "abcd";
    char str4[] = "abcdef";
    printf("strcmp(%s, %s) = %d\n", str1, str2, strcmp(str1, str2));
    printf("my_strcmp(%s, %s) = %d\n", str1, str2, my_strcmp(str1, str2));
    printf("strcmp(%s, %s) = %d\n", str2, str1, strcmp(str2, str1));
    printf("my_strcmp(%s, %s) = %d\n", str1, str2, my_strcmp(str2, str1));
    printf("strcmp(%s, %s) = %d\n", str1, str3, strcmp(str1, str3));
    printf("my_strcmp(%s, %s) = %d\n", str1, str2, my_strcmp(str1, str3));
    printf("strcmp(%s, %s) = %d\n", str1, str4, strcmp(str1, str4));
    printf("my_strcmp(%s, %s) = %d\n", str1, str2, my_strcmp(str1, str4));
    return 0;
}
```

```
strcmp(abcdef, abcdazzzzz) = 1
my_strcmp(abcdef, abcdazzzzz) = 1
strcmp(abcdazzzzz, abcdef) = -1
my_strcmp(abcdef, abcdazzzzz) = -1
strcmp(abcdef, abcd) = 1
my_strcmp(abcdef, abcdazzzzz) = 1
strcmp(abcdef, abcdef) = 0
my_strcmp(abcdef, abcdazzzzz) = 0
请按任意键继续. . .
```

字符串操作5:字符串中字符与子串查找

指针简介:指针是保存**内存地址**的变量，它可以保存某个变量在内存中的地址或者用它**存储某个地址**，指针一般初始化为NULL(0)。

函数char *strchr(const char *s, int c);传入一个字符c，在字符串s中查找**第一次出现**c的地址并返回，若未找到返回NULL。

函数char *strstr(const char *str1, const char *str2);用于判断字符串str2是否是str1的**子串**。如果是，则返回str2在str1中首次出现的地址；否则，返回NULL。

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

int main(){
    char str1[] = "abcabcdazzzzz";
    char str2[] = "abcd";
    char *strx = strchr(str1, 'c');
    char *stry = strstr(str1, str2);
    char *strz = strchr(str1, 'x');
    char *strw = strstr(str1, "vvv");
    printf("strx = %s\n", strx);
    printf("stry = %s\n", stry);
    printf("strz = %s\n", strz);
    printf("strw = %s\n", strw);
    return 0;
}
```

```
strx = cabcdazzzzz
stry = abcdazzzzz
strz = (null)
strw = (null)
请按任意键继续. . .
```

没有结束符的字符串

没有结束符的字符串是**错误的**字符串，如果一个**字符数组**中存储的字符**不包括**结束符'\0'，可以把它当作是字符数组进行使用，但不能把它当作字符串使用，即**不能调用字符串相关函数**进行字符串的相关操作。所以在对字符串进行**构造**时，一定要在最后**添加'\0'**。

```
str1 = 0123456789( strlen = 11
str2 = 01234 strlen = 5
i = 0 [0] [0]
i = 1 [1] [1]
i = 2 [2] [2]
i = 3 [3] [3]
i = 4 [4] [4]
i = 5 [5] [ ]
i = 6 [6] [ ]
i = 7 [7] [ ]
i = 8 [8] [ ]
i = 9 [9] [ ]
请按任意键继续. . .
```

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

int main(){
    char str1[10] = "0123456789";
    char str2[10] = {'0', '1', '2', '3', '4'};
    printf("str1 = %s strlen = %d\n", str1, strlen(str1));
    printf("str2 = %s strlen = %d\n", str2, strlen(str2));
    int i;
    for (i = 0; i < 10; i++){
        printf("i = %d [%c] [%c]\n", i, str1[i], str2[i]);
    }
    return 0;
}
```

课间休息10分钟！

有问题提出！

例2-字符串拆分与排序

问题描述：从键盘读入**任意行**字符串，该字符串会以'#'结束，直到遇到**一个'#'字符**结束读入。将这些字符串中的**单词**提取出来后按照**字典顺序排序**输出。

输入与输出要求：输入的所有字符串中只包含**空格字符**与**小写字符**(‘a’-‘z’)，最大长度不超过100，**以’#’结尾**；输出的单词每个占一行，总共的单词数量不会超过100个，每个单词的长度小于20。

Input Sample:

```
zzz zzzz p#
   abc  ccc  kkk  aaa#
mm word i#
i love coding#
great#
  a  #
#
```

Output Sample:

```
a
aaa
abc
ccc
coding
great
i
i
kkk
love
mm
p
word
zzz
zzzz
```

思考解决该问题需要几个步骤，如何为各部分划分功能并设计函数。

例2-函数与算法设计

需要设计**两个函数**:

1.字符串拆分**提取单词**的函数:

int split_word(const char line[], char word[][MAX_WORD_LEN], int word_cnt);

输入line，即需要提取单词的字符串，将line中的单词**添加**到字符串数组word中，word_cnt是当前字符串数组中**已保存**的单词数量，**返回最终**字符串数组word中的**单词个数**。

例如:

line[]=" abc ccc kkk aaa#"

word[10][MAX_WORD_LEN] = {"zzz", "zzzz", "p"};

word_cnt = 3;

调用函数后:

word[10][MAX_WORD_LEN] = {"zzz", "zzzz", "p", "abc", "ccc", "kkk", "aaa"};

函数返回7。

2.二维的单词数组(字符串数组)**排序**的函数:

void sort_word(char word[][MAX_WORD_LEN], int word_cnt);

对字符串数组word进行排序，排序按照**字典顺序**，字符串数组word中保存了word_cnt个单词。

排序后，word[10][MAX_WORD_LEN] = {"aaa", "abc", "ccc", "kkk", "p", "zzz", "zzzz"};

3.思考其中的**实现算法**。

例2-课堂练习，拆分单词

```
#include <string.h>
#define MAX_LEN 100
#define MAX_WORD_LEN 20

//将line中的单词提取添加到字符串数组word中，
//word_cnt是当前字符串数组中已保存的
//单词数量，返回最终字符串数组word中的单词个数
int split_word(const char line[], char word[][MAX_WORD_LEN], int word_cnt){
    char temp[MAX_WORD_LEN];
    int temp_len = 0;
    int i = 0;
    while(line[i]){
        if(          1          ){
            temp[temp_len] =  2
            temp_len++;
        }
        else{
            if (temp_len){
                          3
                strcpy(   4   , temp);
                word_cnt++;
                          5
            }
        }
        i++;
    }
    return word_cnt;
}
```

line[]=" abc ccc kkk aaa#"

word[MAX_LEN][MAX_WORD_LEN] =
{"zzz", "zzzz", "p"};

word_cnt = 3

word[MAX_LEN][MAX_WORD_LEN] =
{"zzz", "zzzz", "p", "abc", "ccc", "kkk", "aaa"};

3分钟填写代码，

有问题随时提出！

例2-课堂练习，拆分单词，实现

```
#include <string.h>
#define MAX_LEN 100
#define MAX_WORD_LEN 20

//将line中的单词提取添加到字符串数组word中，
//word_cnt是当前字符串数组中已保存的
//单词数量，返回最终字符串数组word中的单词个数
int split_word(const char line[], char word[][MAX_WORD_LEN], int word_cnt){
    char temp[MAX_WORD_LEN];
    int temp_len = 0;
    int i = 0;
    while(line[i]){
        if( line[i] >= 'a' && line[i] <= 'z' ){
            temp[temp_len] = line[i];
            temp_len++;
        }
        else{
            if (temp_len){
                temp[temp_len] = '\0';
                strcpy( word[word_cnt], temp);
                word_cnt++;
                temp_len = 0;
            }
        }
        i++;
    }
    return word_cnt;
}
```

line[]=" abc ccc kkk aaa#"

word[MAX_LEN][MAX_WORD_LEN] =
{"zzz", "zzzz", "p"};

word_cnt = 3

word[MAX_LEN][MAX_WORD_LEN] =
{"zzz", "zzzz", "p", "abc", "ccc", "kkk", "aaa"};

例2-排序与整体代码，实现与测试

```
void sort_word(char word[][MAX_WORD_LEN], int word_cnt){
    int i, j;
    for (i = 0; i < word_cnt; i++){
        for (j = i + 1; j < word_cnt; j++){
            if (strcmp(word[i], word[j]) > 0){
                char temp[MAX_WORD_LEN];
                strcpy(temp, word[i]);
                strcpy(word[i], word[j]);
                strcpy(word[j], temp);
            }
        }
    }
}
int main(){
    char line[MAX_LEN];
    char word[MAX_LEN][MAX_WORD_LEN];
    int word_cnt = 0;
    gets(line);
    while(line[0] != '#'){
        word_cnt = split_word(line, word, word_cnt);
        gets(line);
    }
    sort_word(word, word_cnt);
    int i;
    for (i = 0; i < word_cnt; i++){
        printf("%s\n", word[i]);
    }
    return 0;
}
```

```
zzz zzzz p#
    abc   ccc   kkk   aaa#
mm word i#
i love coding#
great#
   a   #
#
a
aaa
abc
ccc
coding
great
i
i
kkk
love
mm
p
word
zzz
zzzz
请按任意键继续. . .
```

例3-二进制加法

已知使用**两个字符串**表示的**二进制**数，求它们的**和**并返回。注意，由于二进制数用字符串表示，所以可以表示**无穷大**的数，返回的结果使用的内存在**该函数**中开辟，由调用该函数的位置**释放**。

例如:

a = "11"，b = "1"，返回 "100"。

a = "10110"，b = "11110111"，返回 "100001101"。

```
char *addBinary(char *a, char *b) {
        //返回的空间需要在函数中malloc
}
```

选自 **LeetCode 67. Add Binary**

https://leetcode.com/problems/add-binary/description/

难度:**Easy**

例3-思考与分析

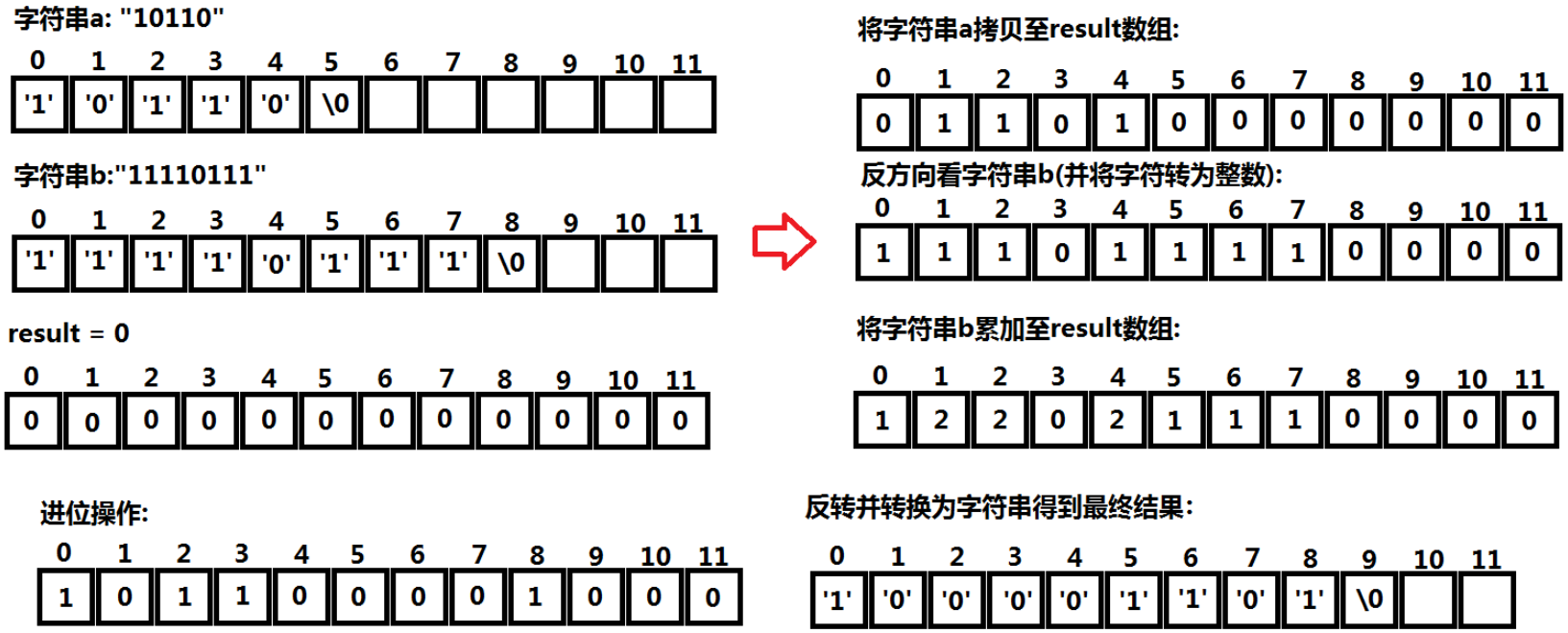
a = "11"，b = "1"，返回 "100"。
a = "10110"，b = "11110111"，返回 "100001101"。
a = “1011001010100000011001010101001000100101010111111111”;
b = “11110111010100101010101010”;
返回 "1011001010100000011001010101100111011111111110101001"。

思考:

1.题目中使用**字符串**表示**二进制**进行加法计算的**原因**是什么，有什么**好处**？
2.可否将字符串形式的二进制转换为**整型**，**完成相加**计算后将结果再转换为**二进制字符串**输出？
3.设计**整体算法**，对二进制整数进行加法计算，它的**时间**与空间复杂度是什么？
4.如果设计**十进制**的**大整数**加法，算法有什么**相同或不同**的地方？

例3-大整数加法，算法设计

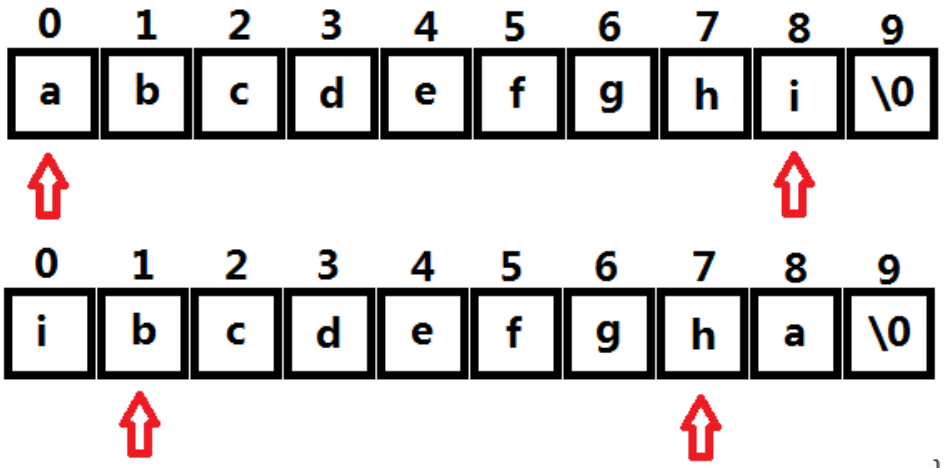
1.设字符串a与字符串b为**待相加**的二进制字符串，result**存储相加后**的二进制结果(整型)。
2.将字符串a反向**拷贝**至result表示的二进制数组中；
3.将字符串b**累加**至result表示的二进制数组中。
4.对result中的结果进行**进位操作**。
5.将result结果**转换**为字符型并**反转**。



例3-字符串反转，课堂练习

使用两个指针分别指向字符串的首尾，

利用两个指针进行字符的交换，直到字符串的中点。



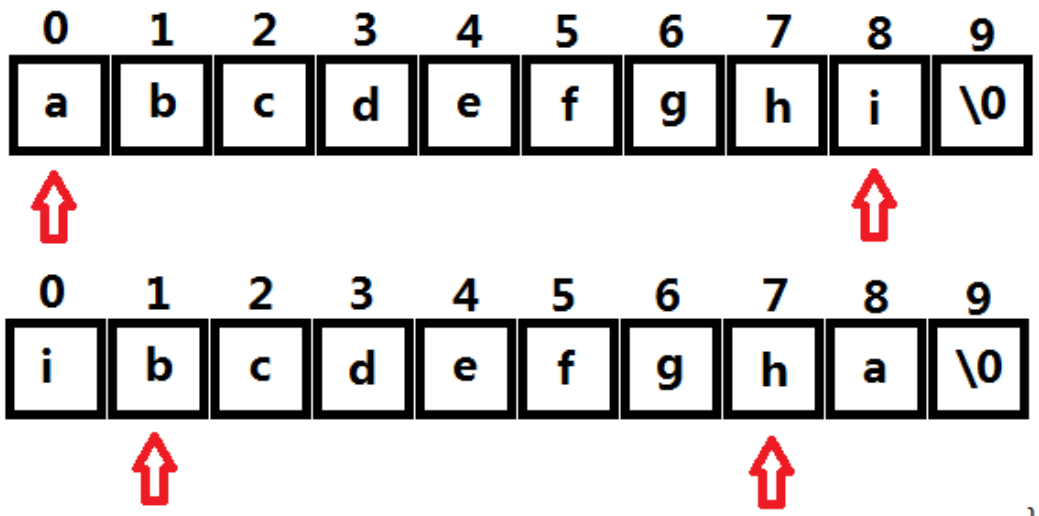

```
#include <string.h>
void string_rev(char *str){
    int len = strlen(str);
    int half_len =
    int i;
    for (i = 0; i < half_len; i++){
        char temp = str[i];
        2
        3
    }
}
```

1

例3-字符串反转，实现

使用两个指针分别指向字符串的首尾，

利用两个指针进行字符的交换，直到字符串的中点。




```
#include <string.h>
void string_rev(char *str){
    int len = strlen(str);
    int half_len = (len + 1) / 2;
    int i;
    for (i = 0; i < half_len; i++){
        char temp = str[i];
        str[i] = str[len - i - 1];
        str[len - i - 1] = temp;
    }
}
```

例3-二进制相加，课堂练习

```
char *addBinary(char *a, char *b) {
    int len1 = strlen(a);  //计算字符串a,b长度，并申请存储结果的空间
    int len2 = strlen(b);
    int max_len = len1 > len2 ? len1 : len2;
    char *result = malloc(max_len + 10);
    memset(result, 0, max_len + 10);
    int i = len1 - 1;
    int j = 0;          //i指向字符串尾部，j指向头部
    while(i >= 0){
        1
        i--;
        j++;        //将字符串a拷贝至result
    }
    i = len2 - 1;
    j = 0;
    while(i >= 0){
        2
        i--;
        j++;    //将字符串b累加至result
    }
    //进位并将整型转为字符型
    for (i = 0; i < max_len; i++){
        result[i + 1] += 3
        result[i] = 4
    }
    if (result[i]){
        5
    }
    string_rev(result);
    return result;
}
```

3分钟填写代码，

有问题随时提出！

例3-二进制相加，实现

```
char *addBinary(char *a, char *b) {
    int len1 = strlen(a);  //计算字符串a,b长度，并申请存储结果的空间
    int len2 = strlen(b);
    int max_len = len1 > len2 ? len1 : len2;
    char *result = malloc(max_len + 10);
    memset(result, 0, max_len + 10);
    int i = len1 - 1;
    int j = 0;          //i指向字符串尾部，j指向头部
    while(i >= 0){
        result[j] = a[i] - '0';
        i--;
        j++;        //将字符串a拷贝至result
    }
    i = len2 - 1;
    j = 0;
    while(i >= 0){
        result[j] += b[i] - '0';
        i--;
        j++;    //将字符串b累加至result
    }
    //进位并将 整型转为字符型
    for (i = 0; i < max_len; i++){
        result[i + 1] += result[i] / 2;
        result[i] = result[i] % 2 + '0';
    }
    if (result[i]){
        result[i] += '0';
    }
    string_rev(result);
    return result;
}
```

例3-测试与leetcode提交结果

```
100001101
10110010101000000110010101011001110111111111010101001
请按任意键继续. . .
```

```
int main(){
    char a1[] = "10110";
    char b1[] = "11110111";
    char *c = addBinary(a1, b1);
    printf("%s\n", c);
    char a2[] = "10110010101000000110010101010010001001010111111111";
    char b2[] = "11110111010100101010";
    c = addBinary(a2, b2);
    printf("%s\n", c);
    free(c);
    return 0;
}
```

Add Binary

Submission Detail

294 / 294 test cases passed. Status: Accepted

Runtime: **0 ms** Submitted: **0 minutes ago**

数组统计字符串中字符数量

使用**数组下标**，统计字符串中的各个**字符**出现的**次数**。

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

int main(){
    int char_map[128] = {0};
    char str[] = "abcdefgaaxxy"; //统计字符串中，各个字符的数量
    int len = strlen(str);
    int i;
    for (i = 0; i < len; i++){
        char_map[str[i]]++;
    }
    for (i = 0; i < 128; i++){
        if (char_map[i] > 0){
            printf("[%c][%d] : %d\n", i, i, char_map[i]);
        }
    }
    return 0;
}
```

//ASC2码 从0 - 127，故使用数组下标做映射，最大范围至128

char_map['a']++;
即char_map[97]++;
char_map['b']++;
即char_map[98]++;
...

```
[a][97] : 3
[b][98] : 1
[c][99] : 1
[d][100] : 1
[e][101] : 1
[f][102] : 1
[g][103] : 1
[x][120] : 2
[y][121] : 1
```

例4:最长回文串

已知一个只包括大小写字符的**字符串**，求用**该字符串中的字符**可以生成的**最长回文字符串**长度。

例如 s = “abcccddaa”，可生成的**最长回文**字符串长度为9，如dccaaaccd、adccbccda、 acdcacdca等，都是**正确**的。

```
int longestPalindrome(char* s) {

}
```

选自 **LeetCode 409. Longest Palindrome**
https://leetcode.com/problems/longest-palindrome/description/
难度:**Easy**

例4:思考

例如在s = “abccccddaa”中，有3个a，1个b，4个c，2个d。

使用字符串s中的字符，**任意组合**，生成新的字符串，若生成的字符串为回文字符串，需要除了**中心**字符，其余字符**只要头部出现，尾部就要对应出现**。

例如:

a…a、ccd…dcc、cc…d…cc

思考:

在字符串中，当遇到某字符数量为**偶数**的字符，我们应该如何处理?

在字符串中，当遇到某字符数量为**奇数**的字符，我们应该如何处理?

例4:分析

例如，在s = “abccccddaa”中，有3个a，1个b，4个c，2个d。

1.在字符串中，字符数量为**偶数**的字符：

全部使用，**头部**放一个字符，**尾部**就对应放一个。

如，4个c和2个d可**全部**用上:

…ccd…dcc…、 …cdc…cdc…、…dcc…ccd…等。

2.在字符串中，字符数量为**奇数**的字符：

丢掉一个字符，**剩下的字符数**为偶数个，按照字符数量为**偶数**的字符处理。

如，3个a中，有2个a可以用上：…a…a…

3.若有**剩余的**字符，如:

1个a、1个b

随便选择1个字符当作中心字符：…a…、…b…

例4:算法设计

1.利用**数组统计字符**方法，**统计**字符串中所有的**字符数量**；

2.设置最长回文串**偶数字符**长度为max_length = 0；

3.设置是否是否有**中心点**标记 flag = 0；

4.**遍历**每一个字符，**字符数**为count，若count为**偶数**，max_length += count；若count为**奇数**，max_length += count – 1，flag = 1；

5.最终最长回文子串长度: max_length + flag。

例如，在s = “abccccddaa”中，有3个a，1个b，4个c，2个d:

1.3个a，max_length += 2; flag = 1;如生成aa;

2.1个b，max_length += 0; 忽略b;

3.4个c，max_length += 4; 如生成ccaacc;

4.2个d，max_length += 2; 如生成dccaaccd;

flag = 1;

故可生成如: dccaaaccd、dccabaccd

最终长度: max_length + flag = 8 + 1 = 9

例4:课堂练习

```
#include <string.h>
int longestPalindrome(char* s) {
    int char_map[128] = {0}; //字符数组统计各字符的数量
    int max_length = 0; //回文串偶数部分最大长度
    int flag = 0; //是否有中心点
    int i;
    int len = strlen(s);

    for (i = 0; i < len; i++){
            1
    }
    for (i = 0; i < 128; i++){
        if (     2     ){
            max_length += char_map[i];
        }
        else{
            3
            flag = 1;
        }
    }
    return max_length + flag;
}       //最终结果是偶数部分长度加上是否有中心点
```

3分钟填写代码，
有问题随时提出！

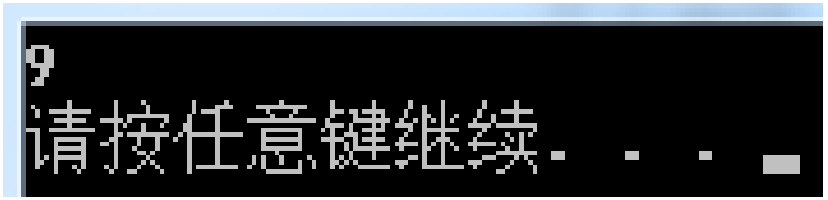
例4:实现

```
#include <string.h>
int longestPalindrome(char* s) {
    int char_map[128] = {0}; //字符数组统计各字符的数量
    int max_length = 0; //回文串偶数部分最大长度
    int flag = 0; //是否有中心点
    int i;
    int len = strlen(s);

    for (i = 0; i < len; i++){
        char_map[s[i]]++;
    }
    for (i = 0; i < 128; i++){
        if (char_map[i] % 2 == 0){
            max_length += char_map[i];
        }
        else{
            max_length += char_map[i] - 1;
            flag = 1;
        }
    }
    return max_length + flag;
}    //最终结果是偶数部分长度加上是否有中心点
```

例4:测试与leetcode提交结果

```
int main(){
    char s[] = "abccccddaa";
    printf("%d\n", longestPalindrome(s));
    return 0;
}
```



Longest Palindrome

Submission Detail

95 / 95 test cases passed. Status: Accepted

Runtime: **0 ms** Submitted: **0 minutes ago**

结束

非常感谢大家！

林沐

联系我们

小象学院：互联网新技术在线教育领航者

– 微信公众号：**小象学院**