程序设计语言与方法(C语言)

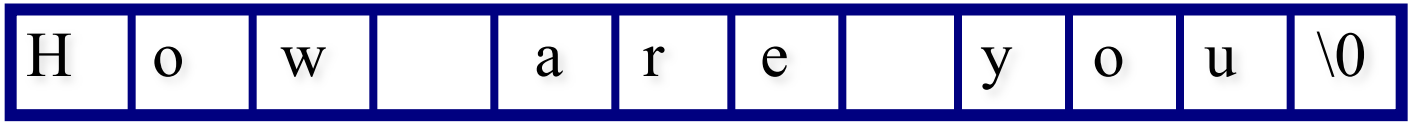
第十章 字符串

字符

- 可打印字符
- 不可打印字符
- ASCII码
- 字符常量与字符变量
  - '0' '1' '@' 'V' \0x30 48
  - char ch1 ='0'; /* ch1 = \0x30; ch1=48; */
- 字符运算
  - 按字符的ASCII码值进行算术运算与逻辑运算
  - Char ch1 = '0';
  - ch1 = ch1 + 1;

字符串

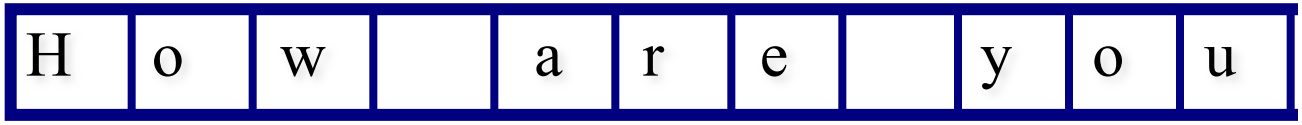
- 一个以字符 '\0' 结尾的字符序列
  - 内存中的形式（按字节顺序存放）



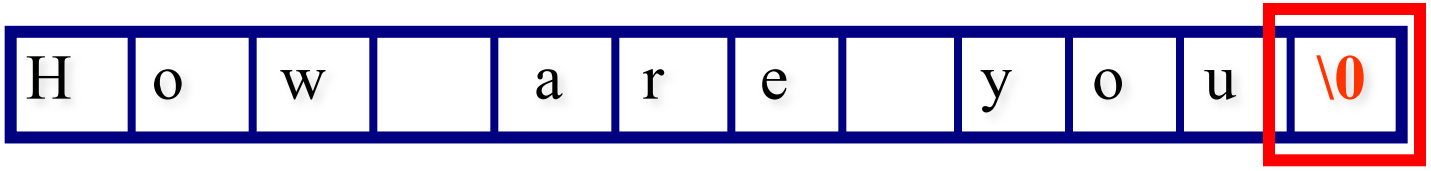
  - 字符串常量
    - 用双引号括起来的一串字符是字符串常量
      - "How are you"
      - C语言会自动为字符串常量添加结束符号'\0'

字符数组与字符串

- 每个元素都是字符类型的数组
- char str[11];



  - 这是一个字符数组，但不代表是字符串
- char str[12];



  - 这是一个字符数组，但也可以代表一个字符串

字符数组的初始化

- 用字符型数据对数组进化初始化
  - char str[6] = {'C','h','i','n','a','\0'};

- 用字符串常量直接初始化数组
  - char str[6] = {"China"};
  - char str[6] = "China";

注意：数组的长度要比实际长度多一个字符位置。

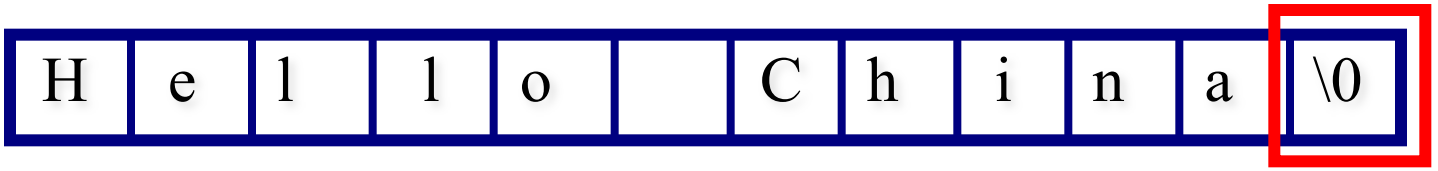
为什么?

字符指针

- C语言并没有为字符串提供任何专门的表示法，完全使用字符数组和字符指针来处理

```
"Hello China"
```

字符串是一串用双引号引起来的字符



数组最后一个元素必须是'\0'才表示字符串

字符数组就是每个元素都是字符型的数组

pStr

字符指针就是指向字符类型数据的指针

字符指针（续）

- 指向字符类型的数据的指针
  - char *pstr = "Hello China";
  - char str[12] = "Hello China";

  - 试一试
    - char str[12];
    - char *pstr;
    - str = "Hello China";　　/* 错误 */
    - pstr = "Hello China";　　/* 正确 */

为什么?

字符串的访问

- 字符数组的访问
  - char str[]= "hello china";
  - char ch;
  - str[0] = 'H';
  - str[5] = '\0';
  - ch = str[5];
- 字符指针的访问
  - char *pstr = "hello china";
  - char ch;
  - *pstr = 'H';
  - *(pstr + 2) = 'l';
  - ch = *(pstr + 3);

字符串的输出

- 按字符逐个输出
- 字符数组 char str[i] = "hello china";
  - for(i = 0; str[i] != '\0'; i++) {
    - printf("%c", str[i]);
  - }
  - printf(“%c”, '\n');
- 字符指针 char *pstr="hello china";
  - for(;*pstr != '\0'; pstr++) {
    - printf("%c", *pstr);
  - }
  - printf("%c", '\n');

```
for(i = 0;*tpstr != '\0'; i++) {
        printf("%c", *(pstr + i));
}
printf("%c", '\n');
```

```
char *tpstr = pstr;
for(;*tpstr != '\0'; tpstr++) {
        printf("%c", *tpstr);
}
printf("%c", '\n');
```

字符串的输入

- 字符串中是否包含有空格字符?
  - 不包含空格字符
    - scanf
  - 包含空格字符
    - gets
      - char * gets(char *s);
      - 调用前，s要指向确定的存储位置（数组或malloc）

空格、回车或制表（Tab）符是输入数据的分隔符，因而不能被读入，输入遇到这些字符时，系统认为字符串输入结束

- 试一试
  - 从键盘输入一个长度不超过12个字符的人名，然后把它输出到屏幕上。
  - 计算实际输入的人名的长度。

字符串函数

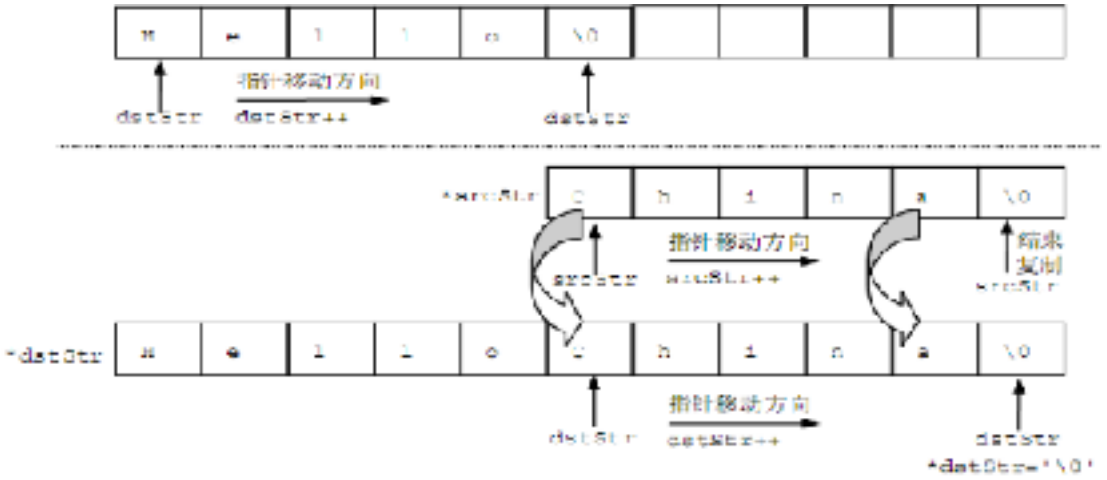
- 头文件名称
  - string.h
- 常用函数
  - strlen(字符串)
    - 求字符串的长度
  - strcpy(目的字符串, 源字符串)
    - 将源字符串拷贝到目的字符串
  - strcat(目的字符串, 源字符串)
    - 将源字符串拼接到目的字符串的后面
  - strcmp(字符串1, 字符串2)
    - 比较两个字符串的大小

常用的字符串函数

函数功能	函数调用的一般形式	功能描述及其说明
求字符串长度	`strlen(str);`	由函数值返回字符串 str 的实际长度，即不包括'\0'在内的实际字符的长度
字符串拷贝	`strcpy(str1,str2);`	将字符串 str2 复制到字符数组 str1 中，这里应确保字符数组 str1 的大小足以存放得下字符串 2
字符串比较	`strcmp(str1,str2);`	比较字符串 str1 和字符串 str2 的大小，结果分为 3 种情况： • 当 str1 大于 str2 时，函数返回值大于 0 • 当 str1 等于 str2 时，函数返回值等于 0 • 当 str1 小于 str2 时，函数返回值小于 0 字符串的比较方法为：对两个字符串从左至右按字符的 ASCII 码值大小逐个字符相比较，直到出现不同的字符或遇到'\0'为止
字符串连接	`strcat(str1,str2);`	将字符串 str2 添加到字符数组 str1 中的字符串的末尾，字符数组 str1 中的字符串结束符被字符串 str2 的第一个字符覆盖，连接后的字符串存放在字符数组 str1 中，函数调用后返回字符数组 str1 的首地址。这里，字符数组 str1 应定义得足够大，以便能存放连接后的字符串
“n 族” 字符串拷贝	`strncpy(str1,str2,n)`	将字符串 str2 的至多前 n 个字符拷贝到字符数组 str1 中
“n 族” 字符串比较	`strncmp(str1,str2,n)`	函数 strncmp(str1, str2, n)的功能与函数 strcmp(str1, str2)类似，它们的不同之处在于，前者最多比较 n 个字符
“n 族” 字符串连接	`strncat(str1,str2,n)`	将字符串 str2 的至多前 n 个字符添加到字符串 str1 的末尾。str1 的字符串结束符被 str2 中的第一个字符覆盖

向函数传递字符串

- 通过字符数组传递
- 通过字符指针传递
- 从函数返回字符串
- 试一试
  - 自己编程实现字符串拼接函数strcat



命令行参数

- int main(int argc, char * argv[]);
- argc
  - 空格分隔的字符串的个数
- argv
  - 命令行中的各个字符串，从左向右序
- 示例
  - PAdd 23 32
    - argc： 3
    - argv
      - argv[0]： "Padd"
      - argv[1]： "23"
      - argv[2]： "32"

命令行参数的处理

- 类型转换
  - 字符串->数值
- 开关特性
  - \i -i --i
- 键值对
  - n=name cnf=\PATHTO\file.txt
- ……