

**课程 实 验 报 告**

**课程名称： C语言程序设计课程实验**

**专业班级： 自实1901**

**学 号： U201916457**

**姓 名： 张皓然**

**指导教师：**

**报告日期： 2020年4月13日**

**人工智能与自动化学院**

# 实验1 字符串在内存中的存储特点？什么叫有效字符串？举例说明数字和数字字符的区别。

## 实验目的

（1）了解字符串的储存特点，了解什么叫做有效字符串。

（2）举例说明数字和数字字符串的区别。

## 设计思路与代码实现

### 设计思路

定义一个字符串，并初始化。然后以不同方式输出。

### 代码实现

1. #include <stdio.h>
2. **void** main()
3. {
4. **char** p = '1';
5. **char** a[20] = "1234";
7. printf("%d %c %s", p,p,a);
8. }

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

49 1 1234

## 实验分析

ASCII码将字符转化为一个整数。如字符‘3’在内存中的值为0x33，即51。字符型变量具有编码后再存储的特点，这使得这种变量需要更多的存储空间。比如10000用字符串型char存储，需要占40位内存。而整形int的10000只需要16位内存。此外，字符型变量也和其他变量一样存在auto，register，static，extern的存储类型，不过实验发现auto在字符型变量中不能再加了。有效字符串即被赋予合理初值的字符串，有效字符串的末尾存放有‘\0’作为字符串结束的标志。

C语言中没有相应的字符串变量，只有字符串常量。我们一般用字符数组或是字符指针来定义字符串。

# 实验2 有哪些数据类型可以处理字符串？各有什么特点？字符串处理输入输出有哪两种方式？一般采用哪一种？用程序进行举例说明

## 实验目的

（1）明确有哪些数据类型可以处理字符串，各有什么特点。

（2）掌握字符串处理输入输出的两种方式。

## 设计思路与代码实现

### 设计思路

用两种方法输入输出一段字符串

### 代码实现

方法一：

1. #include <stdio.h>
2. **int** main()
3. {
4. **char** a[7];
5. scanf("%s", a);
6. printf("%s", a);
7. **return** 0;
8. }

方法二：

1. #include <stdio.h>
2. **int** main()
3. {
4. **char** a[7];
5. gets(a);
6. puts(a);
7. **return** 0;
8. }

## 测试集

张皓然

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

第一种方法：

张皓然

第二种方法：

张皓然

## 实验分析

我们可以用字符数组和字符指针来处理字符串。字符数组是元素为字符型的数组，每一个元素存放一个字符。一个长度为n的字符数组只能存放n-1个字符的字符串，因为在有效字符串的末尾要存一个空字符‘\0’。字符数组可以看作是一个字符串“变量”，可以在后期修改其中字符的内容。但是，形如name=”hello”;（name是已经定义的字符数组）。因为已经经过了定义，这就意味着name已经在内存中有了自己对应的地址。name=”xxxxx”;就相当于把字符串常量”xxxxx”的地址赋给name，这显然对数组是行不通的。

字符指针是一个指针变量。如char\*string=”we are family!”，系统先开辟一块区域存储字符串，然后再将这个字符串的首地址赋予指针。即字符指针指向的是字符串常量，因此他不能修改其中的字符。不过因为字符指针是指针变量，所以它可以指向其他地址。

字符串的输入与输出有两种方式：scanf+printf，gets+puts。我们一般选择后者。注意，puts函数会自动输出一个回车。

# 实验3 举例说明不使用scanf()函数，借助字符串相关库函数完成整型数、浮点数和字符的输入。

## 实验目的

（1）不适用scanf函数，借助字符串相关库函数完成整型数，浮点数，字符的输入。

## 设计思路与代码实现

### 设计思路

先利用gets函数，然后再分别利用atoi，atof函数。

### 代码实现

输入整型数

1. #include <stdio.h>
2. #include<stdlib.h>
3. **int** main()
4. {
5. **char** a[20];
6. **int** A;
7. gets(a);
8. A = atoi(a);
9. printf("%d", A);
10. **return** 0;
11. }

输入浮点数

1. #include <stdio.h>
2. #include<stdlib.h>
3. **int** main()
4. {
5. **char** a[20];
6. **double** A;
7. gets(a);
8. A = atof(a);
9. printf("%f", A);
10. **return** 0;
11. }

输入字符

1. #include <stdio.h>
2. #include<stdlib.h>
3. **int** main()
4. {
5. **char** a[20];
6. gets(a);
7. printf("%c", \*a);
8. **return** 0;
9. }

## 测试集

整型数：10 50 -3 0

浮点数：15.222 16.33 0.0 4.99

字符：a b 1 0 3

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

整型数一组：

10 50 -3 0

浮点数一组：

15.222000 16.330000 0.000000 4.990000

字符一组：

a b 1 0 3

## 实验分析

通过stdlib.h中的atoi，atof，atol函数，就可以将字符串转换为对应类型的数。需要注意两点，第一，atoi等函数放在stdlib.h中，需要引用声明。第二，atof函数的返回值是精度更高的double型函数，虽然函数名是atof，返回的数据却不是float。如果用float型变量接收函数的返回值，有可能导致数据的损失。

# 实验4 字符串的赋值、比较和数值有什么不同？原因是什么？用程序进行举例说明

## 实验目的

（1）掌握字符串的赋值，比较与数值量赋值与比较的区别。。

## 设计思路与代码实现

### 设计思路

分别对字符串，数值，进行赋值，比较的操作。

### 代码实现

数值的赋值与比较

1. #include <stdio.h>
2. **int** main()
3. {
4. **int** a = 1;
5. **int** b = 9;
6. **int** s;
7. s = a;
8. printf("s=%d,(a>b)=%d\n", s, a > b);
9. **return** 0;
10. }

字符串的赋值与比较

1. #include <stdio.h>
2. #include<string.h>
3. **int** main()
4. {
5. **int** s;
6. **char** a[20];
7. strcpy(a, "hello");
8. puts(a);
9. s = strcmp(a, "hellO");
10. printf("%d", s);
11. **return** 0;
12. }

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

数值的赋值与比较：

s=1,(a>b)=0

字符串的赋值与比较：

hello

1

## 实验分析

字符串的赋值利用strcpy(char\*,char\*)函数，它能够将第二个字符串的内容赋值给第一个字符串，函数返回第一个字符串的地址。字符串的比较利用strcmp(char\*,char\*)函数，它将两个字符串的ASCII码逐一比较。若字符串完全一样，函数返回0.如果比较过程中发现某位ASCII不同（第一次发现），就返回1，反之，返回-1.

# 实验5 单个字符串在函数间的传递，需要几个参数？为什么？举例说明

## 实验目的

（1）明确单个字符串在函数间传递所需要的参数个数。

## 设计思路与代码实现

### 设计思路

将字符串传递到函数中。设计一个函数将大写变为小写。

### 代码实现

1. #include <stdio.h>
2. #include<string.h>
3. #include<stdlib.h>
4. **void** strup(**char**\*);
5. **int** main()
6. {
7. **char** a[20];
8. strcpy(a, "hello");
9. strup(a);
10. puts(a);
11. **return** 0;
12. }
13. **void** strup(**char** a[])
14. {
15. **char** t[20];
16. **int** i = 0;
17. **while** (a[i] != '\0')
18. {
19. **if** (a[i] > 'a' && a[i] < 'z')
20. {
21. a[i] -= 0x20;
22. }
23. i++;
24. }
25. }

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

HELLO

## 实验分析

字符串的最后一位是’\0’,实际上已经表明了字符串的位数。所以单个字符串在函数间传递的时候，只需要一个参数——字符串的首地址。

# 实验6 熟悉字符串常用库函数的功能

## 实验目的

（1）熟悉字符串常用库函数的功能。

## 设计思路与代码实现

### 设计思路

字符串常用库函数，

stdio.h：gets，puts

stdlib.h：atoi，atol，atof，itoa

string.h：strcpy，strcat，strcmp，strlen，strupr，strlwr

### 代码实现

stdio.h中的库函数——gets，puts函数的使用

1. #include<stdio.h>
2. **int** main()
3. {
4. **char** a[50];
5. gets(a);
6. puts(a);
7. **return** 0;
8. }

stdlib.h中的库函数——atoi，itoa，atol，atof的使用

1. #include<stdio.h>
2. #include<stdlib.h>
3. **int** main()
4. {
5. **char** a[50];
6. **int** x;
7. gets(a);
8. printf("%d\n", atoi(a));
9. gets(a);
10. printf("%f\n", atof(a));
11. gets(a);
12. printf("%ld\n", atol(a));
13. scanf("%d", &x);
14. itoa(x, a, 10);
15. puts(a);
16. **return** 0;
17. }

string.h中的库函数——strcpy，strcat，strcmp，strlen，strupr，strlwr

1. #include<stdio.h>
2. #include<string.h>
3. **int** main()
4. {
5. **char** a[80];
6. **char** b[80];
7. gets(a);
8. strcpy(b, a);
9. puts(b);
10. strcat(a, b);
11. puts(a);
12. **if** (strcmp(a, b)>0)
13. {
14. puts(a), puts(b);
15. }
16. **else** {
17. puts(b), puts(a);
18. }
19. printf("%d\n", (**int**)strlen(b));
20. strupr(a);
21. strlwr(b);
22. puts(a);
23. puts(b);
24. **return** 0;
25. }

## 测试集

**gets，puts函数的使用：**

第一组：华夏五千年文化，1

第二组：今天的天气很好

第三组：C is very interesting

**atoi，itoa，atol，atof的使用：**

0

120.11

5000000

12

**strcpy等函数的使用：**

第一组：123 56

第二组：abc123%

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

**gets，puts函数的使用**：

第一组：

华夏五千年文化，1

第二组：

今天天气很好

第三组：

C is very interesting

**atoi等函数的使用：**

0

120.110000

5000000

12

**strcpy等函数的使用：**

第一组：

123 56

123 56

123 56123 56

123 56123 56

123 56

6

123 56123 56

123 56

第二组：

abc123%

abc123%

abc123%abc123%

abc123%abc123%

abc123%

7

ABC123%ABC123%

abc123%

# 实验7 编写一个函数完成将8进制的字符串转换为16进制字符串char \*str8to16(char \*s16,char \*s8);另编写main测试函数，完成对char \*str8to16（char \*s16,char \*s8）的测试。

## 实验目的

编写一个函数完成八进制字符串到十六进制字符串的转换。

## 设计思路与代码实现

### 设计思路

先将八进制转换为十进制，再将十进制转换为十六进制。

### 代码实现

1. #include<stdio.h>
2. **char**\* str8to16(**char**\*, **char**\*);//the first char is 8 radix, the second char is 16 radix
3. **int** main()
4. {
5. **char** a[50];     //for 8 radix
6. **char** b[50];     //for 16 radix
7. gets\_s(a, 50);
8. str8to16(a,b);
9. puts(b);
10. **return** 0;
11. }
12. **char**\* str8to16(**char**\* a, **char**\* b)//first trans 8 to 10,then trans 10 to 16
13. {
14. **int** i = 0, j = 0;
15. **long** sum = 0;       //10 radix
16. **int** sixteen[50];
17. a++;                //ignore 0
18. b[0] = '0', b[1] = 'x';//add 0x
19. b += 2;             //ignore 0x
20. **while** (a[i] != '\0')//move to the last of char a
21. {
22. i++;
23. }
24. **while** (j < i)
25. {
26. sum = 8 \* sum + a[j++] - '0';//get 10 radix,for example,0170 >> ((((1)8+7)8)+0 = 120
27. }
28. i = 0, j = 0;
29. **do**              //get 16 redix
30. {
31. **if** (sum % 16 > 9 && sum % 16 < 16)
32. {
33. sixteen[i]=sum % 16 + 87;   //ASCII for a~f
34. }
35. **else**
36. {
37. sixteen[i] = sum % 16 + 48;//ASCII for 0~9
38. }
39. sum /= 16;
40. i++;
41. } **while** (sum != 0);
42. i--;
43. **while** (i >= 0)//put 8 radix into char b
44. {
45. b[j] = sixteen[i];//sixteen[i] is ASCII
46. j++, i--;
47. }
48. b[j] = '\0';//add '\0'to end this char
49. **return** b;
50. }

## 测试集

第一组：0252

第二组：03100

第三组：03421

第四组：01747

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

第一组：0xaa

第二组：0x640

第三组：0x711

第四组：0x3e7

## 实验分析

我们先将8进制字符转换为了10进制的数字。然后通过不断相除以及取余的方法得到了16进制数的每一位。但其实不必这么麻烦，如果我们利用字符串函数itoa，函数的设计将会大大简化。

# 实验8 熟悉库函数strcpy、strcat、strcmp的功能和实现细节。用程序举例说明。

## 实验目的

熟悉strcpy，strcat，strcmp的功能和实现细节。

## 设计思路与代码实现

### 设计思路

自己分别设计出strcpy，strcat，strcmp函数。

### 代码实现

1. #include<stdio.h>
2. **char**\* strcpy(**char**\*, **char**\*);
3. **char**\* strcat(**char**\*, **char**\*);
4. **int** strcmp(**char**\*, **char**\*);
5. **int** main()
6. {
7. **char** a[80], b[80];
8. gets\_s(b);
9. strcpy(a, b);
10. puts(a);
11. strcat(a, b);
12. puts(a);
13. strcmp(a, b);
14. **return** 0;
15. }
16. **char**\* strcpy(**char**\* a, **char**\* b)
17. {
18. **char**\* temp = a;
19. **while** ((\*a++ = \*b++) != '\0')
20. ;
21. **return** temp;
22. }
23. **char**\* strcat(**char**\* a, **char**\* b)
24. {
25. **char**\* temp = a;
26. **while** (\*a != '\0')
27. a++;
28. **while** (\*b != '\0')
29. \*a++ = \*b++;
30. \*a = '\0';
31. **return** temp;
32. }
33. **int** strcmp(**char**\* a, **char**\* b)
34. {
35. **for** (; \*a == \*b && \*a; a++, b++)
36. ;
37. **return**  \*a - \*b;
39. }

## 测试集

hustaia

## 实验过程

（1）编写程序，编译运行（CTRL\_F9）

（2）在用户界面查看运算结果（ALT\_F5）

（3）分析实验结果，总结归纳

## 实验结果

hustaia

hustaiahustaia

## 实验分析

strcpy，strcat，strcmp都是在处理字符串时非常重要的库函数。字符串的相关处理，使用这些函数会非常简洁。

附录

## 开发环境

本次实验中使用的环境配置如下：

（1）操作系统版本：Arch Linux x86\_64

（2）编译器及其版本：GCC version 8.1.1

（3）自动编译工具：CMake version 3.11.4

（4）编程环境：Borland C++ 3.1