字符串与字符指针

// 字符串与字符指针

例：程序func\_array\_pointer6

#include <stdio.h>

int main(int argc, char\* argv[])

{

int i;

char a[] = "I am a boy";

char b[20];

char c[20];

for (i = 0; a[i] != '\0'; ++i)

{

b[i] = a[i];

}

b[i] = '\0'; // 字符串结束标志

for (i = 0; \*(a + i) != '\0'; ++i)

{

\*(c + i) = \*(a + i);

}

\*(c + i) = '\0';

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

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

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

return 0;

}

输出的结果完全相同：

I am a boy

I am a boy

I am a boy

char\* str = "abcde"; // 等同于const char\* str = "abcde";

int i;

for (i = 0; str[i] != '\0'; ++i)

{

printf("%c", str[i]);

}

printf("\n");

for (i = 0; \*(str + i) != '\0'; ++i)

{

printf("%c", \*(str + i));

}

printf("\n");

以上两个循环输出的结果完全相同。

// str[0] = 'p'; // C/C++编译时不会报错，但运行时程序会崩溃

char str1[] = "abcde";

char\* p = NULL;

p = str1;

\*p = 'p';

printf("%s\n", str1);;

输出的结果为：pbcde。

原因如下：

例：程序func\_array\_pointer12

char\* str = "abc"; // 等同于const char\* str = "abc";

// str[0] = 'p'; // error，编译时没问题，运行时程序崩溃

char strArray[4] = "abc";

strArray[2] = 'p';

// 字符串"abc"保存在常量区，里面的内容不允许修改，其

// 本身是常量指针(指向常量的指针)，定义char\* str = "abc";相当于str指向

// 一个字符串常量，所以不能修改其值；而定义char strArray[4] = "abc";

// strArray存放在栈区，所以是将字符指针"abc"的值复制给了strArray数组

// 修改的是存放在栈的字符串"abc"

printf("%s\n", strArray); // abp

例程：字符串的反转

例：程序func\_array\_pointer7

void reverse(char\* str);

int main(int argc, char\* argv[])

{ char str1[] = "abcde";

char\* p = NULL;

int i;

p = str1;

\*p = 'p';

reverse(p);

printf("%s\n", str1);

printf("%s\n", str1);

return 0;

}

void reverse(char\* str)

{

int length = 0;

char\* temp = str;

char c;

if (!str)

{

printf("待反转的字符串无效\n");

return;

}

// 判断字符串的长度

while ((\*str) != '\0')

{

++length;

++str;

}

printf("length = %d\n", length);

str = temp;

temp = str + length - 1;

while (temp > str)

{

c = \*temp;

\*temp = \*str;

\*str = c;

++str;

--temp;

// printf("%c", c);

}

// printf("\n");

}

字符串连接：

调用库函数strcat\_s

char\* str1 = "abc";

char\* str2 = "def";

strcat\_s(str1, 20, str2);

编译时没有问题，但运行时程序崩溃。因为str1指向的是一个常量，不能修改其指向的对象，改成下面的方式：

char str1[20] = "abc";

char\* str2 = "def";

strcat\_s(str1, sizeof(str1), str2);

printf("%s\n", str1);

printf("%s\n", str2);

使用库函数中的strcat\_s函数要注意，指定的尺寸大小必须能将两个字符串完全容纳，否则会报出内存错误。如下所示：

char str1[5] = "abc";

char\* str2 = "def";

strcat\_s(str1, sizeof(str1), str2);

printf("%s\n", str1);

printf("%s\n", str2);

因为两个字符串长度相加大于指定的尺寸5，所以会报出内存错误。

// 字符串连接

例：程序func\_array\_pointer8

// #include <assert.h>

#include <stdio.h>

// #include <string.h>

int Strcat(char\* dest, unsigned int length, char\* src);

int main(int argc, char\* argv[])

{

// char\* str1 = "abc";

char str1[10] = "abc";

char\* str2 = "def";

// char\* str = NULL;

Strcat(str1, sizeof(str1), str2);

// strcat\_s(str1, sizeof(str1), str2);

printf("%s\n", str1);

printf("%s\n", str2);

return 0;

}

int Strcat(char\* dest, unsigned int length, char\* src)

{

unsigned int dest\_length = 0;

unsigned int src\_length = 0;

char\* p = dest;

char\* q = src;

if (!dest)

{

printf("dest指针无效\n");

return 0;

}

if (!src)

{

printf("src指针无效\n");

return 0;

}

// 计算dest的字符串长度

while ((\*p) != '\0')

{

++p;

++dest\_length;

}

// 计算src的字符串长度

while ((\*q) != '\0')

{

++q;

++src\_length;

}

printf("dest length = %d, src length = %d length = %u\n", dest\_length, src\_length, length);

if (length < (dest\_length + src\_length + 1))

{

printf("指定的长度无法容纳dest和src\n");

return 0;

}

while ((\*dest) != '\0')

{

++dest;

}

while ((\*src) != '\0')

{

\*dest = \*src;

++dest;

++src;

}

//while (\*dest++ = \*src++) // 与上一个while循环执行的效果完全相同

//{ // 每次执行赋值后，会进行条件判断，如果\*dest为'\0'，则表示条件为假，会退出while循环。

//}

return 1;

}