指针与字符串

课后练习题

1. 编写一个函数，用于判断给定字符串是否为回文。函数原型如下：

```
int isPalindrome(char *s);
```

其中，s 为给定字符串。若 s 为回文则函数返回 1，否则函数返回 0。

参考程序：

```
int isPalindrome(char *s)
{
    char* start = s;
    char* end = s + strlen(s) - 1;
    for( ; end > start; start++, end--)
        if (*start != *end)
            return 0;
    return 1;
}
```

2. 编写函数实现以下字符串处理功能（**不允许调用 string.h 中提供的函数**）：

（1）用指针操作实现 C 标准库中的 strcat 函数。函数 strcat(s,t)将 t 指向的字符串复制到 s 指向的字符串的尾部。函数原型如下：

```
void strcat(char *s, char *t);
```

说明：假设存储字符串 s 的数组容量充足。

参考程序：

```
void strcat(char *s, char *t)
{
    while (*s)  //s = s + strlen(s);
        s++;
    while (*s++ = *t++)
        ;
}
```

（2）编写函数 strend(s,t)。如果字符串 t 出现在字符串 s 的尾部，该函数返回 1；否则返回 0。函数原型如下：

```
int strend(char *s, char *t);
```

参考程序：

```
int strend(char *s, char *t)
{
    char *bs = s;
    char *bt = t;

    while (*s)  //s = s + strlen(s);
        s++;
    while (*t)  //t = t + strlen(t);
        t++;

    for ( ; *s == *t; s--, t--)
        if (t == bt || s == bs)
            break;

    if (*s == *t && t == bt && *s != '\0')
        return 1;
    return 0;
}
```

（3）编写函数 strindex(s,t)。该函数返回字符串 t 在 s 中**第一次**出现的位置，如果 s 中部不包含 t，则返回-1。函数原型如下：

```
int strindex(char *s, char *t);
```

要求：分别用数组下标操作和指针操作两种方式实现该函数。

扩展：若需要返回 t 在 s 中**最后一次**出现的位置，如何实现？

参考程序：

参考程序 1：用数组下标操作实现基本要求。

```
int strindex(char *s, char *t)
{
    int i, j, k;
    for (i = 0; s[i] != '\0'; i++) {
        for (j=i, k=0; t[k] != '\0' && s[j] == t[k]; j++, k++)
            ;
        if (k > 0 && t[k] == '\0')
            return i;
    }
    return -1;
}
```

参考程序 2：用指针操作实现基本要求。

```
int strindex(char *s, char *t)
{
    char *b = s;
    char *p, *r;

    for ( ; *s != '\0'; s++) {
        for (p=s, r=t; *r != '\0' && *p == *r; p++, r++)
            ;
        if (r > t && *r == '\0')
            return s - b;
    }
    return -1;
}
```

参考程序 3：用数组下标操作实现扩展要求。

```
int strlen(char *s)
{
    int counter = 0;
    while(s[counter])
        counter++;
    return counter;
}

int strindex_last(char *s, char *t)
{
    int i, j, k;
    for (i = strlen(s) - strlen(t); i >= 0; i--) {
        for (j=i, k=0; t[k] != '\0' && s[j] == t[k]; j++, k++)
            ;
        if (k > 0 && t[k] == '\0')
            return i;
    }
    return -1;
}
```

参考程序 4：用指针操作实现扩展要求。

```
int strlen(char *s)
{
    int counter = 0;
    while(*s++)
        counter++;
    return counter;
}

int strindex_last(char *s, char *t)
{
    char *b = s;
    char *p, *r;
    for (s = s + strlen(s) - strlen(t); s != b-1; s--) {
        for (p=s, r=t; *r != '\0' && *p == *r; p++, r++)
            ;
        if (r > t && *r == '\0')
            return s - b;
    }
    return -1;
}
```

3. 写一个函数，将一个整数转化为十进制表示的字符串，函数原型如下：

```
char* itoa(int n, char *string);
```

其中，n 为待转化的整数，string 为用于存储转化结果的字符串。函数返回字符串 string。

要求：采用递归和非递归两种方法实现该函数。

参考程序：

参考程序 1：递归方法

```
char* itoa1 (int n,char *string) {
    if(n < 0) {
        *string++ = '-';
        n = -n;
    }

    if (n / 10)
        string = itoa1(n/10, string);
    *string++ = n % 10 + '0';
    *string = '\0';

    return string;
}
```

```
char* itoa (int n,char *string) {
    itoa1(n, string);
    return string;
}
```

参考程序 2：非递归方法

```
void reverse(char *s)
{
    char c;
    char *t = s + strlen(s) - 1;
    for ( ; s < t; s++, t--) {
        c = *s;
        *s = *t;
        *t = c;
    }
}
char* itoa(int n, char *string)
{
    int sign;
    char *t = string;
    if ((sign = n) < 0)
        n = -n;
    do {
        *string++ = n % 10 + '0';
    } while ((n /= 10) > 0);
    if (sign < 0)
        *string++ = '-';
    *string = '\0';
    reverse(t);
    return t;
}
```

4. 实现字符串的删除和插入操作。

（1）写一个函数用于从一个字符串中删除指定位置指定长度的子串，函数原型如下：

```
char* erase(char *string, char *pstr, int n);
```

其中，string 为待处理的字符串，pstr 为指向字符串 string 内部某个字符或者指向末尾空字符的指针，n 为将被删除的子串的最大长度。函数应实现删除字符串 string 从 pstr 所指向的字符开始（包括）往后的至多 n 个字符，并返回

字符串 string。

说明：如果从 pstr 开始（包括）到字符串的最后一个非空字符（包括）之间有 n 个或 n 个以上字符，则删除从 pstr 指向位置开始的 n 个字符；如果不足 n 个字符，则删除从 pstr 指向位置开始到字符串末尾的所有非空字符。

应用举例：

1） char s1[20] = “Hello World!”;

char *ps = &s1[6];//指向字母‘W’

erase(s1, ps, 2);//执行后，字符串 s1 的内容为“Hello rld!”

2） char s1[20] = “Hello World!”;

char *ps = &s1[6];//指向字母‘W’

erase(s1, ps, 20);//执行后，字符串 s1 的内容为“Hello ”

参考程序：

```
char* erase(char *string, char *pstr, int n)
{
    if (strlen(pstr) < n)
        *pstr = '\0';
    else
        strcpy(pstr, pstr+n);
    return string;
}
```

（2）写一个函数用于将一个字符串插入另一个字符串的指定位置，函数原型如下：

char* insert(char* s1, char* s2, char* pos);

其中，s1 为指向待插入字符串的指针，s2 为指向目标字符串的指针，pos 为指向字符串 s2 内部的某个字符或末尾的空字符的指针。该函数应实现将字符串 s1 中的所有字符（不包括末尾空字符）插入字符串 s2 中 pos 指向的字符之前的位置，并返回字符串 s2。

要求：该函数不得改变字符串 s1。

说明：假设用于存储字符串 s2 的数组容量充足。

应用举例：

char *s1 = “New ”;

char s2[20] = “Hello World!”;

char *ps = &s2[6];//指向字母‘W’

insert(s1, s2, ps);//执行后，字符串 s2 的内容为“Hello New World!”

参考程序：

```
char* insert(char* s1, char* s2, char* pos)
{
    char *p = s2 + strlen(s2);
    char *r = p + strlen(s1);
    while (p != pos-1)
        *r-- = *p--;
    while (*s1)
        *pos++ = *s1++;
    return s2;
}
```