Ch.4 字符串

2017 / 10

原生字符串

- 多个字符
- 以0 ('\0')结尾

怎样表达字符串

- char name[5] = “abcd”;
- char title[] = “president”;
- char *p = “abcdefg”;
- const char * a = “abcd”;

怎样表达字符串

- char name[5] = “abcd”; // 大小为 5
- char title[] = “president”; // 大小为10
- char *p = “abcdefg”; // 大小为 4
- const char * a = “abcd”; // 同上

输出什么结果

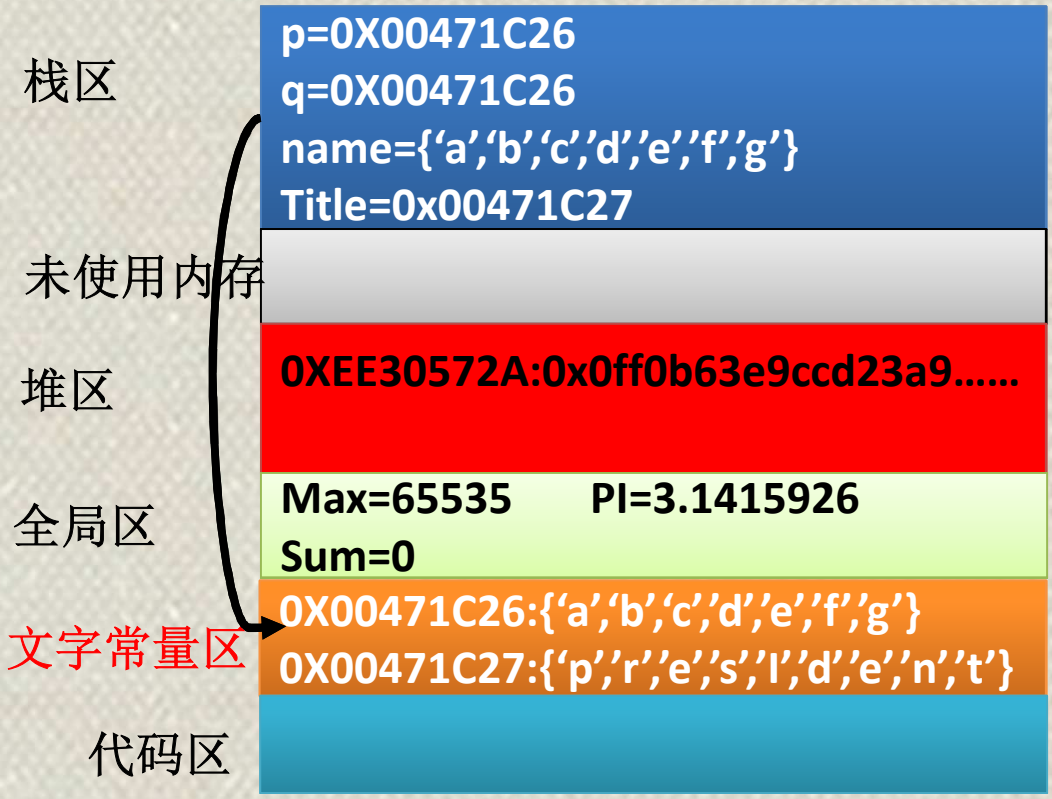
```
char * p = "abcdefg";
char * q = "abcdefg";
const char * r = "abcdefg";
char name[] = "abcdefg";

std::cout << "p:\t" << p << "\tloc:\t" << (int *)p << std::endl;
std::cout << "q:\t" << q << "\tloc:\t" << (int *)q << std::endl;
std::cout << "r:\t" << r << "\tloc:\t" << (int *)r << std::endl;
std::cout << "name:\t" << name << "\tloc:\t" << (int *)name << std::endl;
```

为什么？

- 为什么看起来相同的字符串，地址是相同的？
- 实际的“字符们”存在同一处

存储示意




```
int max = 65535;
const float PI = 3.1415926;
void main()
{
    char * p = "abcdefg";
    char * q = "abcdefg";
    char name [] = "abcdefg";
    char *title = "president";
    static int sum = 0;

    cbyte * dvec = new cbyte[128];
    delete[] dvec;
}
```

字符串常量区

- char * p = “abcdefg”;
- const char * q = “abcdef”;
- p[3] = ‘t’;
- q[3] = ‘t’;
- char name[] = “abcd”;
- p = name;
- p[3] = ‘t’;

常量

- const char * a = “abcd”;
- char * const p = “abcd”;
- const char * const p = “abcd”;

const 就近

- 离哪个近就是修饰哪个的

先研究一下const

- const的作用非常多

深入理解字符串

- 先了解一下字符 ASCII码
- ‘a’、 ‘A’、’ ’、’\t’、’\n’、
- ’\’’ 表达的是 ‘
- ’\\’ 表达的是 \

一些字符串方面的库函数

- strlen // 字符串长度
- strcpy // 字符串拷贝，废弃
- srcat // 字符串连接，废弃
- strstr // 字符串子串，废弃
- strok // 字符串截取，废弃
- strcmp // 字符串比较
- ……

可使用的替代方案

- strncpy_s
- strncat_s
- strok_s

Unicode

- 一个字节表达的字符有限

std::string

- 是 char * 类字符串的封装
- 更方便易用
  - 两个字符串拼接起来， a + b;
- 更面向对象
  - str.length()对比 strlen(str);

```
std::string a = "my ";
std::string b("chinese");
std::string c = a + b + " name";
```

一些使用方法

length/size

substr

find	查找
rfind	反向查找
find_first_of	查找包含子串中的任何字符，返回第一个位置
find_first_not_of	查找不包含子串中的任何字符，返回第一个位置
find_last_of	查找包含子串中的任何字符，返回最后一个位置
find_last_not_of	查找不包含子串中的任何字符，返回最后一个位置
substr	得到字串
compare	比较字符串

……

实现一个trim

- 将字符串中指定的字符串（或字符）剔除

```
void trim(std::string & s, std::string sep)
{
        if (s.empty())
                return;

        size_t pos = s.find_first_of(sep);
        while (pos != std::string::npos)
        {
                s.erase(pos, 1);
                pos = s.find_first_of(sep);
        }
}
```