问题

sizeof 与 strlen 的区别与使用详解。这算是 C++ 中基础中的基础了，很容易被考到。

区别

1. `strlen` 是一个函数，只能以 `char*` (字符串)作为参数，用来计算指定字符串`str` 的长度，但不包括结束字符`'\0'`。所以其参数必须是以`'\0'`作为结束符才可以正确统计其字符长度，否则是个随机数，具体看下面的代码。
2. `sizeof` 是一个单目运算符，它的参数可以是数组、指针、字符串、对象等等，计算的是参数所对应内存空间的实际字节数。
3. 在统计字符串 `str` 的长度时，包含结束字符 `'\0'`

具体看下面的代码进一步理解：

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    char* s1 = "0123456789";
    cout<<sizeof(s1)<<endl;  // 输出 8，因为这时的参数 s 是一个指向字符串常量的字符指针，因此计算的是指针的大小，注意这里不同编译器得到的值可能不同，也有可能是4
    cout<<sizeof(*s1)<<endl;  // 输出 1，*s 是第一个字符
    cout<<strlen(s1)<<endl;  // 输出 10，有10个字符，strlen是个函数，内部实现是用一个循环计算到\0之前为止
    //strlen(*s1);  // 报错，因为strlen函数的参数类型只能是 char* 即字符串

    char s2[] = "0123456789";  // 动态数组
    cout<<sizeof(s2)<<endl;  // 结果为11，数组名虽然本质上是一个指针，但是作为sizeof的参数时，计算的是整个数组的大小，这点要特别注意。且在求动态数组的大小时，sizeof统计到第一个结束字符'\0'处结束
    cout<<strlen(s2)<<endl;  // 结果为10
    cout<<sizeof(*s2)<<endl;  // 结果为1，*s是第一个字符

    char s3[100] = "0123456789";
    cout<<sizeof(s3)<<endl;  // 结果为100，因为内存给数组 s3分配了字节数为100的空间大小
    cout<<strlen(s3)<<endl;  // 结果为10

    int s4[100] = {0,1,2,3,4,5,6,7,8,9};
    cout<<sizeof(s4)<<endl;  // 结果为400，因为int数组中每个元素都是int型，int型占用4字节
    //strlen(s4);  // 报错，strlen不能以int* 作为函数参数

    char p[] = {'a', 'b','c','d','e', 'f', 'g','h'};
    char q[] = {'a', 'b','c','d','\0', 'e', 'f', 'g'};
    cout<<sizeof(p)<<endl;  // 结果为8
    cout<<strlen(p)<<endl;  // 结果是一个随机数，因为字符串数组中没有结束字符 '\0'，因此该函数会一直统计下去，直到碰到内存中的结束字符
    cout<<sizeof(q)<<endl;  // 结果还是8
    cout<<strlen(q)<<endl;  // 结果为4, 结束字符 '\0'前有4个字符
    return 0;
}
```

另外`sizeof`在**统计结构体的大小时还有一个内存对齐的问题**，具体如下：

```
struct Stu {
    int i;
    int j;
    char k;
};

Stu stu;
cout<<sizeof(stu)<<endl;  // 输出 12
```

这个例子是结构体的内存对齐所导致的，计算结构变量的大小就必须讨论数据对齐问题。为了CPU存取的速度最快（这同CPU取数操作有关，详细的介绍可以参考一些计算机原理方面的书），C语言在处理数据时经常把结构变量中的成员的大小按照4或8的倍数计算，这就叫**数据对齐**（data alignment）。这样做可能会浪费一些内存，但理论上速度快了。当然这样的设置会在读写一些别的应用程序生成的数据文件或交换数据时带来不便。

参考资料

sizeof和strlen的区别及使用详解