问题

strcpy 、strncpy 和 memcpy 的区别。这个可能不太常考，但是也是一个易错点，顺便总结下吧。

strcpy

函数原型：`char *strcpy(char *dest, const char *src)`

函数功能：把 src 地址开始且包括结束符的字符串复制到以 dest 开始的地址空间，返回指向 dest 的指针。需要注意的是，src 和 dest 所指内存区域不可以重叠且 dest 必须需有足够的空间来容纳 src 的字符串，strcpy 只用于字符串复制。

安全性：strcpy 是不安全的，strcpy 在遇到结束符时才会正常的结束运行，会因为 src 长于 dest 而造成 dest 栈空间溢出以致于崩溃异常，它的结果未定，可能会改变程序中其他部分的内存的数据，导致程序数据错误，不建议使用。

函数实现

```
char* strcpy(char* dest,const char* src)//src到dest的复制
{
    if(dest == nullptr || src == nullptr)
        return nullptr;
    char* strdest = dest;
    while((*strdest++ = *src++) != '\0') {};
    return strdest;
}
```

下面用代码来感受下：

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char src1[10] = "hello";
    strcpy(src1, src1+1);
    cout<<"src1:"<<src1<<endl;  // 输出 ello。

    char src2[10] = "hello";
    strcpy(src2+1, src2);
    cout<<"src2:"<<src2<<endl;  // 输出 hhello，按照内存重叠逻辑理解，应该输出hhhhh……，后面是随机两才对，因为'\0'被覆盖，而strcpy要遇到'\0'才会停止复制。
                                // 可能对于内存重叠的问题，每种编译器的定义不一样
    char src3[10] = "hello";
    char dest3[3];
    strcpy(dest3, src3);
    cout<<"dest3:"<<dest3<<endl;  // 输出 hello，非常奇怪的是居然没报错，dest3的空间不是比src3的小吗？

    // 注意下面这个用例体现了 strcpy 与 strncpy 的区别
    char *src4 = "best";
    char dest4[30] = "you are the best one.";
```

```
    strcpy(dest4+8, src4);
    cout<<"dest4:"<<dest4<<endl;  // 输出 you are best。字符串最后一个字节存放的是一个空字符—"\0"，用来表示字符串的结束。
                                  // 把src4复制到dest4之后，src4中的空字符会把把复制后的字符串隔断，所以会显示到best就会结束。
    return 0;
}
```

strncpy

函数原型：`char* strncpy(char* dest,const char* src,size_t n)`

函数功能：将字符串 src 中最多 n 个字符复制到字符数组 dest 中(它并不像 strcpy 一样只有遇到 NULL 才停止复制，而是多了一个条件停止，就是说如果复制到第 n 个字符还未遇到 NULL，也一样停止），返回指向 dest 的指针。只适用于字符串拷贝。如果 src 指向的数组是一个比n短的字符串，则在 dest 定义的数组后面补 '\0' 字符，直到写入了n个字符。

注意：如果 n>dest 串长度，dest 栈空间溢出产生崩溃异常。一般情况下，使用 strncpy 时，建议将n置为dest串长度，复制完毕后，为保险起见，将dest串最后一字符置NULL。

安全性：比较安全，当dest的长度小于n时，会抛出异常。

函数实现：

```
char *strncpy(char *dest, const char *src, int len)
{
    assert(dest!=NULL && src!=NULL);
    char *temp;
    temp = dest;
    for(int i =0;*src!='\0' && i<len; i++,temp++,src++)
        *temp = *src;
    *temp = '\0';
    return dest;
}
```

如果想把一个字符串的一部分复制到另一个字符串的某个位置，显然strcpy()函数是满足不了这个功能的，因为strcpy()遇到结束字符才停止。但是 strncpy可以。

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char *src5 = "best";
    char dest5[30] = "you are the best one.";
    strncpy(dest5+8, src5, strlen(src5));
    cout<<"dest5:"<<dest5<<endl;  // 输出 you are bestbest one，注意这里与上面代码最后一个示例的区别
    return 0;
}
```

memcpy

函数原型：`void* memcpy (void* dest, const void* src, size_t n)`

函数功能：与strncpy类似，不过这里提供了一般内存的复制，即memcpy对于需要复制的内容没有任何限制，可以复制任意内容，因此，用途广泛。

函数实现

```
void *memcpy(void *memTo, const void *memFrom, size_t size)
{
    if((memTo == NULL) || (memFrom == NULL)) //memTo和memFrom必须有效

        return NULL;
    char *tempFrom = (char *)memFrom;              //保存memFrom首地址
    char *tempTo = (char *)memTo;                  //保存memTo首地址

    while(size-- > 0)                 //循环size次，复制memFrom的值到memTo中

        *tempTo++ = *tempFrom++ ;
    return memTo;
}
```

注意：memcpy没有考虑内存重叠的情况，所以如果两者内存重叠，会出现错误。

```
#include <iostream>
#include <cstring>
using namespace std;

int main() {
    char *src6 = "best";
    memcpy(src6+1, src6, 3);  // 报错，内存重叠
    cout<<"src6:"<<src6<<endl;
    return 0;
}
```

三者的区别

主要说一下strcpy和memcpy的区别，主要有以下3方面的区别。

1. 复制的内容不同。strcpy只能复制字符串，而memcpy可以复制任意内容，例如字符数组、整型、结构体、类等。
2. 复制的方法不同。strcpy不需要指定长度，它遇到被复制字符的串结束符"\0"才结束，所以容易溢出。memcpy则是根据其第3个参数决定复制的长度。
3. 用途不同。通常在复制字符串时用strcpy，而需要复制其他类型数据时则一般用memcpy

使用情况

1. dest指向的空间要足够拷贝；使用strcpy时，dest指向的空间要大于等于src指向的空间；使用strncpy或memcpy时，dest指向的空间要大于或等于n。
2. 使用strncpy或memcpy时，n应该大于strlen(src)，或者说最好n >= strlen(s1)+1；这个1 就是最后的“\0”。
3. 使用strncpy时，确保dest的最后一个字符是“\0”。

参考资料

strcpy、strncpy和memcpy的用法与区别详解
strcpy()的注意事项以及strncpy()的用处