如下代码的运行结果:

```
#include <iostream>
#include<iomanip>
using namespace std;
#include<string.h>
int main(int argc, char* argv[])
{
	char str1[] = "hello world";
	char str2[] = "hello world";
	int size1 = sizeof(str1);
	//str1 的字节数
	int length2 = strlen(str2);
	//str2 的长度
	const char* str3 = "hello world";
	const char* str4 = "hello world";
	int size3 = sizeof(str3);
	int length4 = strlen(str4);
	//str4 的长度
	//
	//str1 和 str2 不相等
	if(str1 == str2)
	{
		std::cout<<"str1 and str2 are same."<<std::endl;
	}
	else
	{
		std::cout<<"str1 and str2 are not same."<<std::endl;
	}
	if(str3 == str4)
	{
		std::cout<<"str3 and str4 are same."<<std::endl;
	}
	else
	{
		std::cout<<"str3 and str4 are not same."<<std::endl;
	}

std::cout<<"str1_size:"<<setw(4)<<size1<<std::endl<<"str3_size:"<<setw(4)<<size3<<std::
endl;

std::cout<<"str2_length:"<<setw(4)<<length2<<std::endl<<"str4_length:"<<setw(4)<<lengt
```

```
h4<<std::endl;
    return 0;
}
```

str1 和 str2 是两个字符数组,我们会为它分配两个长度为 12 字节单位的空间,并把 "hello world" 复制到数组当中去.这是两个初始地址不同的数组,因此 str1 和 str2 数组的值也不尽相同

str3 和 str4 是两个指针,我们无需给它分配内存以存储字符串的内容,而只需把它们指向 "hello world" 在内存当中的地址就可以了.由于 "hello world" 是常量字符串,他在内存中只有一个拷贝.所以比较 str3 和 str4 的值也是相同的,输出的第二行是 "str3 and str4 are same."

char arr[ ] = "abcdef"; sizeof(arr) = 4/8; strlen(arr) = 7; char* str = "hello world"; *str; sizeof(str) = 4/8; strlen(str) = 12;

字符串操作:

>> : 字符串复制函数: char * my_strcpy(char * dest, const char * src);

```
char *my_strcpy(char *dest, const char *src)
{
    char *tmp = dest;
    while(*dest++ = *src++)
    {
        ;
    }
    return tmp;
}
```

>> : 字符串拼接函数: char *my_strcat(char *dest, char *src);

```
char *my_strcat(char *dest, char *src)
{char *temp = dest;
    while(*dest)
    {
        dest++;
    }
    while(*dest++ = *src++)
    {
        ;
    }
    return temp;
}
```

>> : 实现 memcpy 拷贝:内存拷贝: void * memcpy(void * dest, const void * src, size_t len);

```
void *memcpy(void *dst, const void *src, size_t len)
{
	if (NULL == dst || NULL == src)
	{
		return NULL;
	}
	void *ret = dst;
	if (dst <= src || (char *)dst >= (char *)src + len)
	{
	//没有内存重叠，从低地址开始复制
	while (len--)
	{
		*(char *)dst = *(char *)src;
		dst = (char *)dst + 1;
		src = (char *)src + 1;
	}
	}
	else
	{
		//有内存重叠，从高地址开始复制
		src = (char *)src + len - 1;
		dst = (char *)dst + len - 1;
		while (len--)
		{
			*(char *)dst = *(char *)src;
		dst = (char *)dst - 1;
		src = (char *)src - 1;
		}
	}
	return ret;
}
```

>> : 实现 my_strlen 字符串长度函数: char * my_strlen(const char * str);

```
int my_strlen(const char *str)
{
	if(*str == '\0')
	{
		return 0;
	}
	else
	{
		return 1 + my_strlen(str + 1);
```

```
    }
}
```

>> : 字符串比较函数:my_strcmp； int strcmp(char *str1, char *str2);

```
/*next 数组下标表示字符串长度，因此初始化要多初始化一个位置*/
void compute_next(char *s, int *next)
{
    int i = 0, len = strlen(s);
    int k = 0;

    next[0] = next[1] = 0;
    for(i = 1; i < len; i++)
    {
        while(k > 0 && s[k] != s[i])
            k = next[k];
        if(s[k] == s[i])
            k++;
        next[i+1] = k;
    }
}

char *kmp_strstr(char *s, char *t)
{
    int s_len, t_len, i, j=0;
    int *next = NULL;

    if(!s || !t)
        return NULL;

    s_len = strlen(s);
    t_len = strlen(t);
    if(t_len == 0)
        return s;
    next = (int *)malloc(sizeof(int) * (t_len + 1));
    if(!next)
        return NULL;

    compute_next(t, next);

    for(i = 0; i < s_len; i++)
    {
        while(j > 0 && s[i] != t[j])
            j = next[j];
        if(s[i] == t[j])
```

```
                j++;
            if(j == t_len)
            {
                //printf("match %d\n", i-j+1);
                free(next);
                return s+i-j+1;
            }
        }

        free(next);
        return NULL;
}
```

>> : 查找一个字串:my_strstr(const char * str1, const char * str2);

```
#include <stdio.h>
/*
 //在字符串 str1 中查找字符串 str2 出现的位置并返回一个指针
 */
char* my_strstr(char const* str1, char const* str2)
{
        register char* last;
        register char* current;
        /*
        把指针初始化成我们已经找到的前一次匹配的位置.
        */
        last = NULL;
        /*
          只在第二个字符串 str2 不为空时查找
         */
        if(*str2 != '\0')
        {
            current = strstr(str1, str2);
            /*
            我们每次找到字符串时，让指针指向它的起始位置.然后查找该字符串下一个匹配
的位置.
            */
            while(current != NULL)
            {
                last = current;
                current = strstr(last + 1, str2);
            }
        }
        return last;        //返回指向我们最后一次匹配到起始位置的指针
}
```

```
int main()
{
	printf("Hello world\n");
	return 0;
}
```
