

B1029066 傅作君

正確:

```
#include<stdio.h>
```

```
#include<stdlib.h>
```

```
char *my_strcpy(char * , const char * );
```

```
int main()
```

```
{
```

```
    char src[] = "cs23!";
```

```
    char dst[]="Hello hello";
```

```
    char *curdst;
```

```
    int len=0;
```

```
    printf("src address %p and first char %c \n", (void *)&src, src[0]);
```

```
    printf("dst address %p and first char %c \n", (void *)&dst, dst[0]);
```

```
    // compute where NULL character is '\0' ASCII 0
```

```
    while(src[++len]);
```

```
    // print out the char arrays and various addresses.
```

```
    printf("src array %s and last element %d\n", src, atoi(&src[len]));
```

```
    printf("dst array %s and last element %c\n", dst, dst[len]);
```

```
    // do the copy
```

```
    curdst= my_strcpy(dst, src);
```

```
    // check to see if the NULL char is copied too.
```

```
    printf("dst array %s and last element %d\n", dst, atoi(&dst[len]));
```

```
    return 0;
```

```
}
```

```
char *my_strcpy(char *s1, const char *s2) {  
  
    register char *d = s1;  
  
    // print the pointer variables address and their contents, and first char  
  
    printf("s2 address %p, its contents is a pointer %p to first char %c \n",  
          (void *)&s2, (void *)s2, *s2);  
    printf("s1 address %p, its contents is a pointer %p to first char %c \n",  
          (void *)&s1, (void *)s1, *s1);  
  
    while ((*d++ = *s2++));  
    return(s1);  
}
```

}錯誤:

```
#include<stdio.h>  
#include<stdlib.h>
```

```
char *my_strcpy(char * , const char * );
```

```
int main()  
{
```

```
    char src[] = "cs23!";  
    char dst[]="Hello hello";  
    char *curdst;  
    int len=0;
```

```
    printf("src address %p and first char %c \n", (void *)&src, src[0]);  
    printf("dst address %p and first char %c \n", (void *)&dst, dst[0]);
```

```
    // compute where NULL character is '\0' ASCII 0
```

```
    while(src[len++]);
```

```
    // print out the char arrays and various addresses.
```

```

printf("src array %s and last element %d\n", src, atoi(&src[len]));
printf("dst array %s and last element %c\n", dst, dst[len]);

// do the copy

curdst= my_strcpy(dst, src);

// check to see if the NULL char is copied too.

printf("dst array %s and last element %d\n", dst, atoi(&dst[len]));

return 0;

}

char *my_strcpy(char *s1, const char *s2) {

    register char *d = s1;

    // print the pointer variables address and their contents, and first char

    printf("s2 address %p, its contents is a pointer %p to first char %c\n",
           (void *)&s2, (void *)s2, *s2);
    printf("s1 address %p, its contents is a pointer %p to first char %c\n",
           (void *)&s1, (void *)s1, *s1);

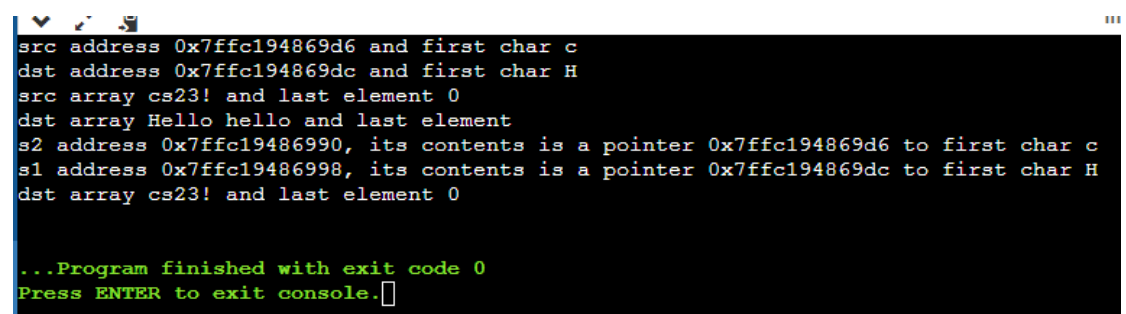
    while ((*d++ = *s2++));

    return(s1);

}

```

正確的執行結果:



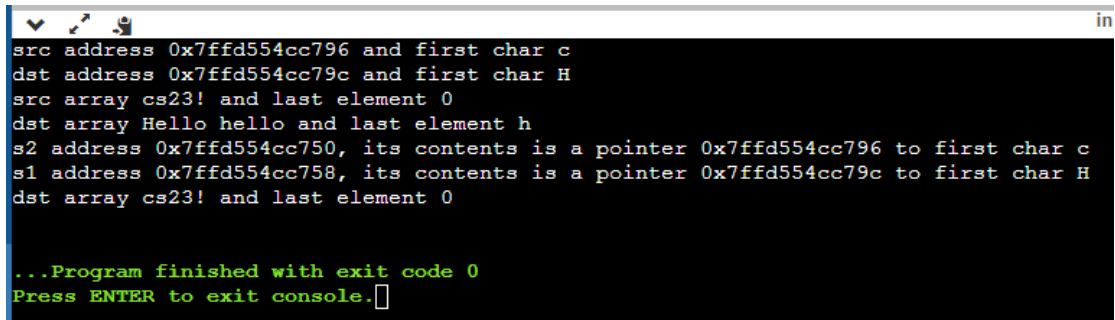
```

src address 0x7ffc194869d6 and first char c
dst address 0x7ffc194869dc and first char H
src array cs23! and last element 0
dst array Hello hello and last element
s2 address 0x7ffc19486990, its contents is a pointer 0x7ffc194869d6 to first char c
s1 address 0x7ffc19486998, its contents is a pointer 0x7ffc194869dc to first char H
dst array cs23! and last element 0

...Program finished with exit code 0
Press ENTER to exit console.

```

錯誤的執行結果:

A screenshot of a console window with a black background and white text. The window has a title bar with standard OS icons on the left and the text 'in' on the right. The output text is as follows:

```
src address 0x7ffd554cc796 and first char c
dst address 0x7ffd554cc79c and first char H
src array cs23! and last element 0
dst array Hello hello and last element h
s2 address 0x7ffd554cc750, its contents is a pointer 0x7ffd554cc796 to first char c
s1 address 0x7ffd554cc758, its contents is a pointer 0x7ffd554cc79c to first char H
dst array cs23! and last element 0

...Program finished with exit code 0
Press ENTER to exit console.
```

這個程式碼的目的在於把 **src** 字串複製到 **dst** 字串，看到正確的執行結果，**src** 陣列的最後一個元素(也就是第五個元素)是 **/0**，對應到 **dst** 陣列的第五個元素應該要是一個空格；錯誤的執行結果則顯示為 **h**。

原因在於 **while** 迴圈當中對於 **src** 字串長度的計算，錯誤者使用 **len++**，這樣會讓值先加一再執行，導致 **/0** 算完後仍然加一並執行才終止長度計算，讓 **dst** 跟著多計算了一位而跑到空格後面的 **h** 元素；正確者使用 **++len**，此為先執行才加一的寫法，讓 **src** 字串長度能夠正確的運算並正確對應到 **dst** 字串的元素(**/0** 對應到空格)。