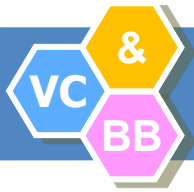


NHẬP MÔN LẬP TRÌNH

CHUỖI KÝ TỰ





Nội dung

1

Khái niệm

2

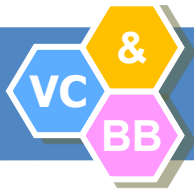
Khởi tạo

3

Các thao tác trên chuỗi ký tự

4

Bài tập



Khái niệm

❖ Khái niệm

- Kiểu **char** chỉ chứa được một ký tự. Để lưu trữ một chuỗi (nhiều ký tự) ta sử dụng mảng (một chiều) các ký tự.
- Chuỗi ký tự kết thúc bằng ký tự '**\0**' (null)
→ Độ dài chuỗi = kích thước mảng – 1

❖ Ví dụ

```
char hoten[30];    // Dài 29 ký tự  
char ngaysinh[9]; // Dài 8 ký tự
```



❖ Khởi tạo như mảng thông thường

▪ Độ dài cụ thể

```
char s[10] = { 'T', 'H', 'C', 'S', ' ', 'A', '\0' };  
char s[10] = "THCS A"; // Tự động thêm '\0'
```

0 1 2 3 4 5 6 7 8 9

'T' 'H' 'C' 'S' ' ' 'A' '\0' [] [] []

▪ Tự xác định độ dài

```
char s[] = { 'T', 'H', 'C', 'S', ' ', 'A', '\0' };  
char s[] = "THCS A"; // Tự động thêm '\0'
```

0 1 2 3 4 5 6

'T' 'H' 'C' 'S' ' ' 'A' '\0'



Xuất chuỗi

❖ Sử dụng hàm printf với đặc tả “%s”

```
char monhoc[50] = "Tin hoc co so A";  
printf("%s", monhoc);    // Không xuống dòng
```

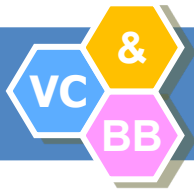
```
Tin hoc co so A_
```

❖ Sử dụng hàm puts

```
char monhoc[50] = "Tin hoc co so A";  
puts(monhoc);    // Tự động xuống dòng  
⇔ printf("%s\n", monhoc);
```

```
Tin hoc co so A
```

```
_
```



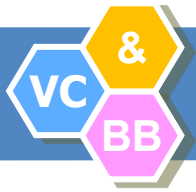
Nhập chuỗi

❖ Sử dụng hàm scanf với đặc tả “%s”

- Chỉ nhận các ký tự từ bàn phím đến khi gặp ký tự khoảng trắng hoặc ký tự xuống dòng.
- Chuỗi nhận được không bao gồm ký tự khoảng trắng và xuống dòng.

```
char monhoc[50];  
printf("Nhap mot chuoai: ");  
scanf("%s", monhoc);  
printf("Chuoai nhan duoc la: %s", monhoc);
```

```
Nhap mot chuoai: Tin hoc co so A  
Chuoai nhan duoc la: Tin_
```



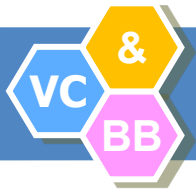
Nhập chuỗi

❖ Sử dụng hàm gets

- Nhận các ký tự từ bàn phím đến khi gặp ký tự xuống dòng.
- Chuỗi nhận được là những gì người dùng nhập (trừ ký tự xuống dòng).

```
char monhoc[50];  
printf("Nhap mot chuoai: ");  
gets(monhoc);  
printf("Chuoai nhan duoc la: %s", monhoc);
```

```
Nhap mot chuoai: Tin hoc co so A  
Chuoai nhan duoc la: Tin hoc co so A _
```



Một số hàm thao tác trên chuỗi

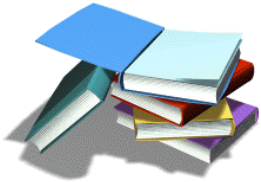
❖ Thuộc thư viện `<string.h>`

- strcpy
- strdup
- strlwr/strupr
- strrev
- strcmp/stricmp
- strcat
- strlen
- strstr



Hàm sao chép chuỗi

`char *strcpy(char dest[], const char src[])`



Sao chép chuỗi **src** sang chuỗi **dest**, dừng khi ký tự kết thúc chuỗi **'\0'** vừa được chép.
! dest phải đủ lớn để chứa src

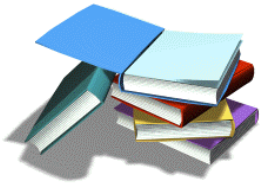


◆ Địa chỉ chuỗi dest



```
char s[100];  
s = "Tin học cơ sở A";           // sai  
strcpy(s, "Tin học cơ sở A");    // đúng
```

```
char *strdup(const char s[])
```



Tạo bản sao của một chuỗi s cho trước.
Hàm sẽ tự tạo vùng nhớ đủ chứa chuỗi s.



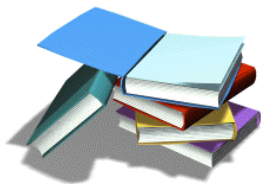
- ◆Thành công: Địa chỉ chuỗi kết quả
- ◆Thất bại: null



```
char *s;  
s = strdup("Tin hoc co so A");
```

Hàm chuyển chuỗi thành chữ thường

`char *strlwr(char *s)`



Chuyển chuỗi s thành chuỗi thường ('A' thành 'a', 'B' thành 'b', ..., 'Z' thành 'z')



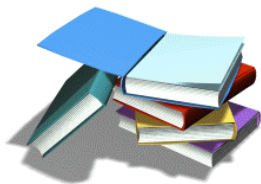
◆ Địa chỉ chuỗi s



```
char s[] = "Tin hoc co so A!!!";  
strlwr(s);  
puts(s);           // tin hoc co so a!!!
```

Hàm chuyển chuỗi thành chữ IN

`char *strupr(char *s)`



Chuyển chuỗi s thành chuỗi in ('a' thành 'A', 'b' thành 'B', ..., 'z' thành 'Z')



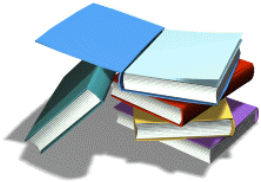
◆ Địa chỉ chuỗi s



```
char s[] = "Tin hoc co so A!!!";  
strupr(s);  
puts(s);           // TIN HOC CO SO A!!!
```

Hàm đảo ngược chuỗi

`char *strrev(char *s)`



Đảo ngược thứ tự các ký tự trong chuỗi (trừ ký tự kết thúc chuỗi)



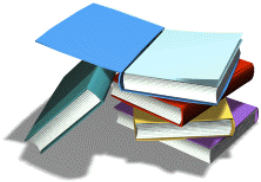
◆ Địa chỉ chuỗi kết quả



```
char s[] = "Tin hoc co so A!!!";  
strrev(s);  
puts(s);           // !!!A os oc coh niT
```

Hàm so sánh hai chuỗi

int **strcmp**(const char ***s1**, const char ***s2**)



So sánh hai chuỗi s1 và s2 (phân biệt hoa thường)



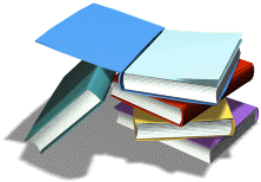
- ◆ < 0 nếu s1 < s2
- ◆ == 0 nếu s1 == s2
- ◆ > 0 nếu s1 > s2



```
char s1[] = "tin hoc co so A!!!";  
char s2[] = "hoc tin co so A!!!";  
int kq = strcmp(s1, s2); // => kq > 0
```

Hàm so sánh hai chuỗi

int stricmp(const char ***s1**, const char ***s2**)



So sánh hai chuỗi s1 và s2 (không phân biệt hoa thường)



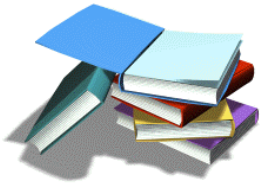
- ◆ < 0 nếu s1 < s2
- ◆ == 0 nếu s1 == s2
- ◆ > 0 nếu s1 > s2



```
char s1[] = "tin hoc co so A!!!";  
char s2[] = "TIN HOC CO SO A!!!";  
int kq = stricmp(s1, s2); // => kq == 0
```

Hàm nối hai chuỗi

`char* strcat(char *dest, const char *src)`



Nối chuỗi src vào sau chuỗi dest.
! Chuỗi dest phải đủ chứa kết quả



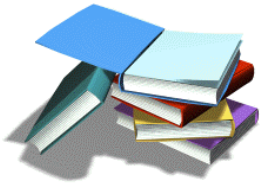
◆ Địa chỉ của chuỗi được nối



```
char s1[100] = "Tin hoc";  
char s2[] = "co so A!!!";  
strcat(s1, " "); // => "Tin hoc "  
strcat(s1, s2);  // => "Tin hoc co so A!!!"
```


Hàm tính độ dài chuỗi

`size_t* strlen(const char *s)`



Tính độ dài chuỗi s
`size_t` thay cho unsigned (trong `<stddef.h>`)
dùng để đo các đại lượng không dấu.



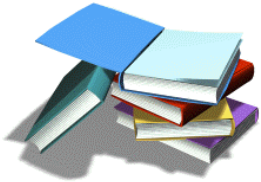
◆ Độ dài chuỗi s



```
char s[] = "Tin hoc co so A!!!";  
int len = strlen(s);           // => 18
```

Hàm tìm chuỗi trong chuỗi

char* strstr(const char *s1, const char *s2)



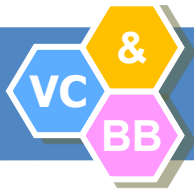
Tìm vị trí xuất hiện đầu tiên của s2 trong s1



- ◆Thành công: trả về con trỏ đến vị trí xuất hiện đầu tiên của s2 trong s1.
- ◆Thất bại: trả về null

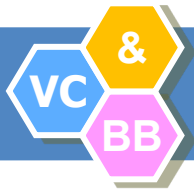


```
char s1[] = "Tin hoc co so A!!!";  
char s2[] = "hoc";  
if (strstr(s1, s2) != null)  
    printf("Tim thay!");
```



Bài tập

- ❖ **Bài 1:** Xem thêm một số hàm khác như
 - atoi, atol, atof : đổi chuỗi thành số
 - itoa, ltoa, ultoa: đổi số thành chuỗi
 - strtok
- ❖ **Bài 2:** Viết hàm upper(char s[]) đổi toàn bộ các ký tự sang ký tự hoa (giống hàmstrupr)
- ❖ **Bài 3:** Viết hàm lower(char s[]) đổi toàn bộ các ký tự sang ký tự thường (giống hàmstrlwr)
- ❖ **Bài 4:** Viết hàm proper(char s[]) đổi các ký tự đầu tiên của mỗi từ sang ký tự hoa.



Bài tập

- ❖ **Bài 5:** Viết hàm `standard(char s[])` bỏ toàn bộ khoảng trắng đầu chuỗi, cuối chuỗi và giữa 2 từ trong `s` chỉ còn 1 khoảng trắng.
- ❖ **Bài 6:** Xóa tất cả các khoảng trắng của `s`
- ❖ **Bài 7:** Đếm xem có bao nhiêu từ trong `s`. Xuất các từ trên các dòng liên tiếp.
- ❖ **Bài 8:** Tìm từ có chiều dài dài nhất và in ra.
- ❖ **Bài 9:** Trích ra `n` ký tự đầu tiên/cuối cùng/bắt đầu tại vị trí `pos`.

