



# Chương 8

## String Type (Chuỗi ký tự)

### Nội dung

1

Khái niệm

2

Khai báo

3

Khởi tạo

4

Thao tác trên chuỗi ký tự

5

Bài tập

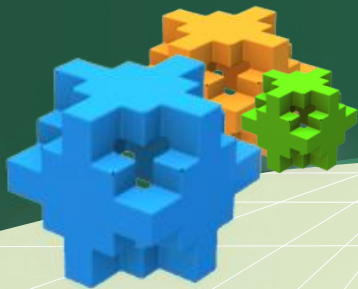




# String Type (Chuỗi ký tự)

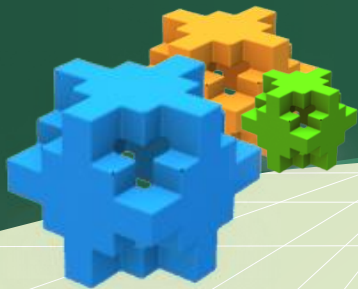
## Giới thiệu

- ❖ Nội dung trong chương này:
  - Giới thiệu về Chuỗi
  - Giới thiệu về cách truy xuất Chuỗi
- ❖ Bạn sẽ có thể:
  - Thực hiện các thao tác trên chuỗi trong ngôn ngữ C/C++



# 1. Khái niệm

- String hay còn gọi là xâu kí tự, chuỗi kí tự, ... là một tập hợp kí tự. Chúng hình thành nên câu, từ, đoạn văn...
- C/C++ hỗ trợ hai phương pháp biểu diễn dữ liệu string:
  - Phương pháp biểu diễn bằng mảng char của C (C-style string).
  - Phương pháp biểu diễn bằng class string mới trong C++ (C++-Style string).



## 2. Khai báo

### ■ Khai báo: char

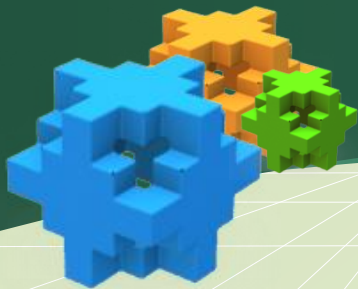
- Khái niệm: Kiểu char chỉ chứa được một ký tự. Để lưu trữ một chuỗi (nhiều ký tự) ta sử dụng mảng(một chiều) các ký tự.
- Chuỗi ký tự kết thúc bằng ký tự '\0' (NULL).
- Cú pháp:

Kiểu dữ liệu chuỗi tên chuỗi [độ dài chuỗi];

Ví dụ:

```
char Hoten[30]; // chuỗi chứa đc 29 ký tự
```

```
char chucvu[9]; // chuỗi chứa đc 8 ký tự
```



## 2. Khai báo

### ❖ Khai báo chuỗi sử dụng class string

✓ C++ cung cấp class/kiểu dữ liệu string để quản lý chuỗi ký tự. Để sử dụng class string, ta cần phải khai báo thư viện:  
`#include <string>`

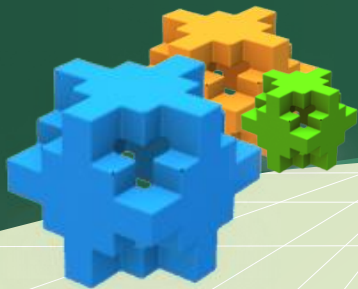
✓ Ưu điểm của class string là khả năng sử dụng các toán tử.

Ví dụ:

Sử dụng toán tử + để ghép hai string lại với nhau thay vì phải sử dụng hàm như phương pháp mảng char.

❖ Lưu ý rằng mảng char không thể được áp dụng các toán tử với nhau như string, mà phải có giá trị kiểu string trong phép toán





## 2. Khai báo

- ❖ Khai báo chuỗi sử dụng class string

Cú pháp:

```
string <string name>;
```

Ví dụ: `string chuoikt;`



### 3. Khởi tạo

A. Khởi tạo như mảng thông thường:

- Độ dài cụ thể:

Char s[10]={ 'L', 'A', 'C', ' ', 'H', 'O', 'N', 'G', '\0' }

Char s[10]="LAC HONG" // tự động thêm ký tự \0

0	1	2	3	4	5	6	7	8	9
'L',	'A'	'C'	' '	'H'	'O'	'N'	'G'	'\0'	



### 3. Khởi tạo

A. Khởi tạo như mảng thông thường:

- Tự xác định độ dài:

```
Char s[]={'L','A','C',' ','H','O','N','G','\0'}
```

```
Char s[]="LAC HONG" // tự động thêm ký tự \0
```

0	1	2	3	4	5	6	7	8
'L'	'A'	'C'	' '	'H'	'O'	'N'	'G'	'\0'





### 3. Khởi tạo

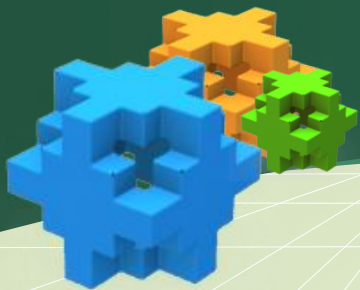
B. Khởi tạo sử dụng string:

```
string s="KHOA CNTT";
```

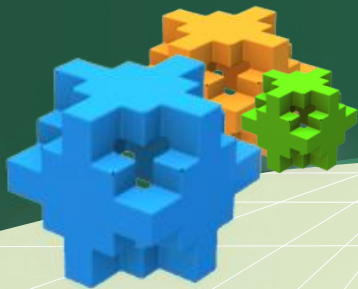
HOẶC:

```
string s("KHOA CNTT");
```

0	1	2	3	4	5	6	7	8	9
'K'	'H'	'O'	'A'	' '	'C'	'N'	'T'	'T'	'\0'

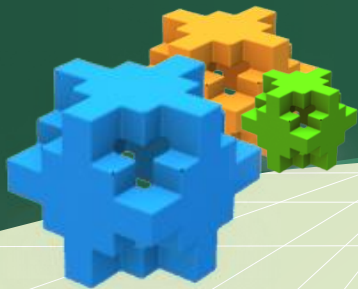


## 4. Các thao tác trên chuỗi ký tự



## 4. Các thao tác trên chuỗi ký tự

- a) Nhập chuỗi (Input Strings)
- b) Xuất chuỗi (Output Strings)
- c) Các hàm xử lý chuỗi



## 4.1 Xuất chuỗi:

a) Sử dụng hàm printf

```
Char MonHoc[20]="Ngon Ngu C";  
printf("%s",MonHoc);
```

```
1 #include<stdio.h>  
2 void main()  
3 {  
4 char monhoc[20]="ngon ngu c";  
5 printf("%s",monhoc);  
6 }  
7  
8 "C:\Users\CPU\Documents\C-Free\Temp\Untitled10.exe"  
ngon ngu cPress any key to continue . . .
```



## 4.1 Xuất chuỗi:

### 4.1 Xuất chuỗi:

b) Sử dụng hàm cout:

```
Char MonHoc[20]="Ngon Ngu C";  
cout<<MonHoc;
```

```
ngon ngu c  
Press any key to continue . . .
```





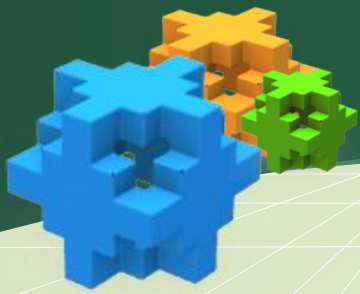
## 4.1 Xuất chuỗi:

### 4.1 Xuất chuỗi:

c) Sử dụng hàm put:

```
Char MonHoc[20]="Ngon Ngu C";  
puts(monhoc);
```

```
ngon ngu c  
Press any key to continue . . .
```



## 4.1 Xuất chuỗi:

- Một số lưu ý khi xuất chuỗi:

Escape sequence	Ý nghĩa
\\	Kí tự \
\'	Kí tự ngoặc đơn '
\"	Kí tự ngoặc kép "
\?	Dấu hỏi ?
\n	Kí tự xuống dòng



## 4.2 Nhập chuỗi:

### 4.2.1 Nhập mảng ký tự chuỗi:

#### a) Sử dụng hàm scanf với đặc tả “%s”:

- chỉ nhận các ký tự từ bàn phím đến khi gặp ký tự ‘ ‘ hoặc ký tự ‘\n’.
- chuỗi nhận được không bao gồm ký tự ‘ ‘ và ký tự ‘\n’.

```
Char MonHoc[20] = “Ngon Ngu C”;
```

```
Scanf(“%s”, MonHoc);
```

```
printf(“chuoi nhan duoc la %s\n”, MonHoc);
```

```
Ngon Ngu C
chuoi nhan duoc la Ngon
Press any key to continue . . .
```



## 4.2 Nhập chuỗi:

b) Sử dụng hàm gets:

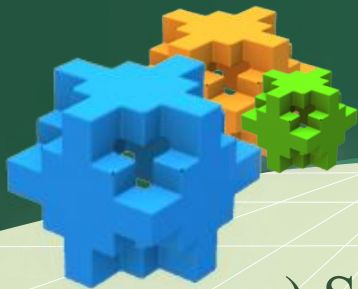
- Nhận các ký tự từ bàn phím đến khi gặp ký tự xuống dòng.
- Chuỗi nhận được là những gì người dùng nhập (trừ ký tự xuống dòng)

```
Char MonHoc[20]="Ngon Ngu C";
```

```
gets("%s",MonHoc);
```

```
cout<<"\n ban vua nhap:"<<s;
```

```
Ngon Ngu C
chuoi nhan duoc la Ngon Ngu C
Press any key to continue . . .
```



## 4.2 Nhập chuỗi:

c) Sử dụng hàm `cin.getline`:

- thuộc lớp namespace `std`. Có chức năng nhận các thông tin từ bàn phím qua hàm ( `std::cin` ) sau đó chuyển vào biến chỉ định

```
Char MonHoc[20]="Ngon Ngu C";
```

```
cin.getline(s,20);
```

```
cout<<"\n ban vua nhap:"<<s;
```

```
Ngon Ngu C
chuoi nhan duoc la Ngon Ngu C
Press any key to continue . . .
```





## 4.2 Nhập chuỗi:

### 4.2.2 Nhập chuỗi- lớp String

- Sử dụng hàm getline:

Khi sử dụng phải khai báo thư viện string  
cấu trúc :

```
getline(std::cin,<bien>);
```

Định nghĩa : thuộc lớp namespace std. Có chức năng nhận các thông tin từ bàn phím qua hàm ( std::cin ) sau đó chuyển vào biến chỉ định.



## 4.2 Nhập chuỗi:

❖ ví dụ:

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string ten;
    cin.ignore(); // thêm vào để xóa bộ nhớ đệm, tránh bị trôi lệnh
    cout<<"Nhập ten của bạn : "<<endl;
    getline(std::cin,ten);
    cout<<"xin chào bạn "<<ten<<" đến với ngôn ngữ lập trình c/c++!!!"<<endl;
    return 0;
}
```

```
VIET NAM
xin chào bạn đến với ngôn ngữ lập trình c/c++!!!
Press any key to continue . . .
```



## 4.2 Nhập chuỗi:

### ❖ Lưu ý:

- Lời khuyên : hàm getline sử dụng dựa trên việc thông qua lưu trữ giá trị hàm cin nhận vào. Ở đây cin chỉ nhận từng lần nhập liệu sau đó sẽ kết thúc khi gặp khoảng trắng hay xuống hàng. vì thế getline lưu trữ toàn bộ thông tin và truyền vào biến.
- Lưu ý : Việc sử dụng hàm getline có thể gây mất biến khi nhấn enter nghĩa là nếu bạn tạo 1 biến có giá trị số sau đó bạn getline và in 1 chuỗi ký tự.
- Nên xóa bộ nhớ đệm trước khi getline để tránh lỗi.

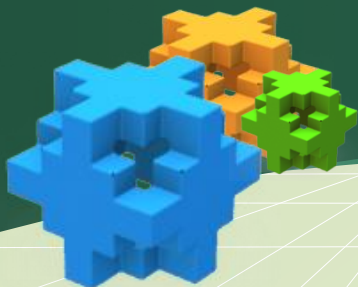


## 4.2 Nhập chuỗi:

❖ Lưu ý:

**std::fflush(stdin);**

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string ten;
    cin.ignore();
    cout<<"Nhap ten cua ban : "<<endl;
    getline(std::cin,ten);
    std::fflush(stdin);
    cout<<"xin chao ban "<<ten<<" den voi
    ngon ngu lap trinh c/c++!!!"<<endl;
    return 0;
}
```



## 4.2 Nhập chuỗi:

Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol	Code	Symbol
0	NUL (null)	20	DC4 (data control 4)	40	(	60	<	80	P	100	d	120	x
1	SOH (start of header)	21	NAK (negative acknowledge)	41	)	61	=	81	Q	101	e	121	y
2	STX (start of text)	22	SYN (synchronous idle)	42	*	62	>	82	R	102	f	122	z
3	ETX (end of text)	23	ETB (end of transmission block)	43	+	63	?	83	S	103	g	123	{
4	EOT (end of transmission)	24	CAN (cancel)	44	,	64	@	84	T	104	h	124	
5	ENQ (enquiry)	25	EM (end of medium)	45	-	65	A	85	U	105	i	125	}
6	ACK (acknowledge)	26	SUB (substitute)	46	.	66	B	86	V	106	j	126	~
7	BEL (bell)	27	ESC (escape)	47	/	67	C	87	W	107	k	127	DEL (delete)
8	BS (backspace)	28	FS (file separator)	48	0	68	D	88	X	108	l		
9	HT (horizontal tab)	29	GS (group separator)	49	1	69	E	89	Y	109	m		
10	LF (line feed/new line)	30	RS (record separator)	50	2	70	F	90	Z	110	n		
11	VT (vertical tab)	31	US (unit separator)	51	3	71	G	91	[	111	o		
12	FF (form feed / new page)	32	(space)	52	4	72	H	92	\	112	p		
13	CR (carriage return)	33	!	53	5	73	I	93	]	113	q		
14	SO (shift out)	34	"	54	6	74	J	94	^	114	r		
15	SI (shift in)	35	#	55	7	75	K	95	_	115	s		
16	DLE (data link escape)	36	\$	56	8	76	L	96	`	116	t		
17	DC1 (data control 1)	37	%	57	9	77	M	97	a	117	u		
18	DC2 (data control 2)	38	&	58	:	78	N	98	b	118	v		
19	DC3 (data control 3)	39	'	59	;	79	O	99	c	119	w		





## 4.3 Hàm xử lý trên chuỗi:

### ❖ 4.3.1 Một số hàm thao tác trên mảng ký tự:

- Hàm strlen

Xác định độ dài chuỗi - Hàm strlen()

Cú pháp: `int strlen(const char* s)`

- Hàmstrupr()

Đổi chuỗi chữ thường thành chuỗi chữ hoa,

Hàmstrupr() được dùng để chuyển đổi chuỗi chữ thường thành chuỗi chữ hoa, kết quả trả về của hàm là một con trỏ chỉ đến địa chỉ chuỗi được chuyển đổi.

Cú pháp: `char *strupr(char *s)`



## 4.3 Hàm xử lý trên chuỗi:

- Hàm `strlwr()`

Đổi chuỗi chữ hoa thành chuỗi chữ thường,

Muốn chuyển đổi chuỗi chữ hoa thành chuỗi toàn chữ thường, ta sử dụng hàm `strlwr()`, các tham số của hàm tương tự như hàm `strupr()`

Cú pháp: `char *strlwr(char *s)`

- Hàm `strcpy()`

Sao chép chuỗi hàm này được dùng để sao chép toàn bộ nội dung của chuỗi nguồn vào chuỗi đích.

Cú pháp: `char *strcpy(char *Des, const char *Source)`



## 4.3 Hàm xử lý chuỗi:

- Hàm strcmp()

Để so sánh hai chuỗi theo từng ký tự trong bảng mã Ascii

Cú pháp: `int strcmp(const char *s1, const char *s2)`

Hai chuỗi s1 và s2 được so sánh với nhau, kết quả trả về là một số nguyên

- Nếu kết quả là số âm, chuỗi s1 nhỏ hơn chuỗi s2.
- Nếu kết quả là 0, hai chuỗi bằng nhau.
- Nếu kết quả là số dương, chuỗi s1 lớn hơn chuỗi s2.



## 4.3 Hàm xử lý chuỗi:

- Hàm strcmp()

Để so sánh hai chuỗi theo từng ký tự trong bảng mã Ascii

Cú pháp: `int strcmp(const char *s1, const char *s2)`

Hai chuỗi s1 và s2 được so sánh với nhau, kết quả trả về là một số nguyên

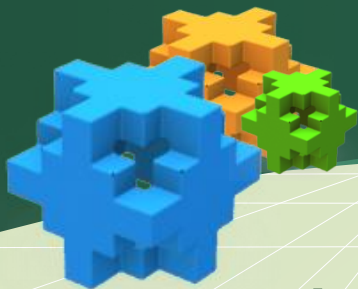
- Nếu kết quả là số âm, chuỗi s1 nhỏ hơn chuỗi s2.
- Nếu kết quả là 0, hai chuỗi bằng nhau.
- Nếu kết quả là số dương, chuỗi s1 lớn hơn chuỗi s2.



## 4.3 Hàm xử lý chuỗi:

- Hàm strrev:
  - Công dụng: đảo ngược chuỗi kí tự
  - Cấu trúc: `char *strrev(char *s);`

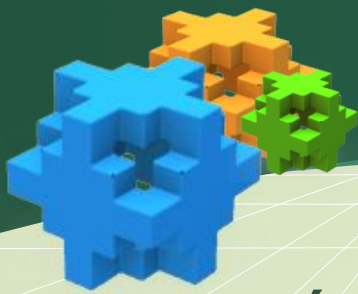




## 4.3 Hàm xử lý trên chuỗi:

### ❖ Các hàm thao tác với kí tự:

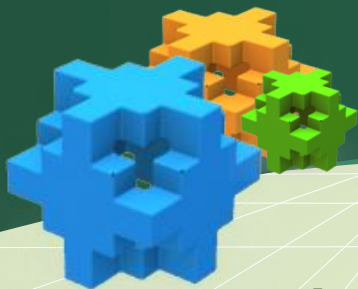
FUNCTION	DESCRIPTION	EXAMPLE
<code>toupper(Char_Exp)</code>	Returns the uppercase version of <i>Char_Exp</i> (as a value of type <code>int</code> ).	<pre>char c = toupper('a'); cout &lt;&lt; c; Outputs: A</pre>
<code>tolower(Char_Exp)</code>	Returns the lowercase version of <i>Char_Exp</i> (as a value of type <code>int</code> ).	<pre>char c = tolower('A'); cout &lt;&lt; c; Outputs: a</pre>
<code>isupper(Char_Exp)</code>	Returns true provided <i>Char_Exp</i> is an uppercase letter; otherwise, returns false.	<pre>if (isupper(c))     cout &lt;&lt; "Is uppercase." else     cout &lt;&lt; "Is not uppercase."</pre>



## 4.3 Hàm xử lý trên chuỗi:

### ❖ Các hàm thao tác với kí tự:

FUNCTION	DESCRIPTION	EXAMPLE
<code>islower(Char_Exp)</code>	Returns true provided <i>Char_Exp</i> is a lowercase letter; otherwise, returns false.	<pre>char c = 'a'; if (islower(c))     cout &lt;&lt; c &lt;&lt; " is lowercase."; <b>Outputs:</b> a is lowercase.</pre>
<code>isalpha(Char_Exp)</code>	Returns true provided <i>Char_Exp</i> is a letter of the alphabet; otherwise, returns false.	<pre>char c = '\$'; if (isalpha(c))     cout &lt;&lt; "Is a letter."; else     cout &lt;&lt; "Is not a letter."; <b>Outputs:</b> Is not a letter.</pre>
<code>isdigit(Char_Exp)</code>	Returns true provided <i>Char_Exp</i> is one of the digits '0' through '9'; otherwise, returns false.	<pre>if (isdigit('3'))     cout &lt;&lt; "It's a digit."; else     cout &lt;&lt; "It's not a digit."; <b>Outputs:</b> It's a digit.</pre>
<code>isalnum(Char_Exp)</code>	Returns true provided <i>Char_Exp</i> is either a letter or a digit; otherwise, returns false.	<pre>if (isalnum('3') &amp;&amp; isalnum('a'))     cout &lt;&lt; "Both alphanumeric."; else     cout &lt;&lt; "One or more are not."; <b>Outputs:</b> Both alphanumeric.</pre>



## 4.3 Hàm xử lý trên chuỗi:

### ❖ Các hàm thao tác với kí tự:

`isspace(Char_Exp)`

Returns true provided *Char\_Exp* is a whitespace character, such as the blank or newline character; otherwise, returns false.

```
//Skips over one "word" and sets c  
//equal to the first whitespace  
//character after the "word":  
do  
{  
    cin.get(c);  
} while (! isspace(c));
```

`ispunct(Char_Exp)`

Returns true provided *Char\_Exp* is a printing character other than whitespace, a digit, or a letter; otherwise, returns false.

```
if (ispunct('?'))  
    cout << "Is punctuation.";  
else  
    cout << "Not punctuation.";
```

`isprint(Char_Exp)`

Returns true provided *Char\_Exp* is a printing character; otherwise, returns false.

`isgraph(Char_Exp)`

Returns true provided *Char\_Exp* is a printing character other than whitespace; otherwise, returns false.

`isctrl(Char_Exp)`

Returns true provided *Char\_Exp* is a control character; otherwise, returns false.



## 4.3 Hàm xử lý trên chuỗi:

### 4.3.2 Xử lý string:

- Hàm `length()` (Độ dài chuỗi):

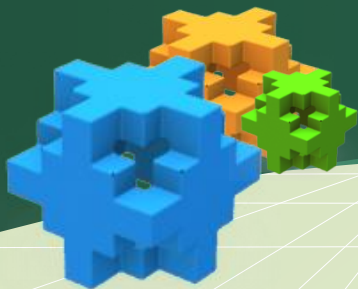
`length()` là một trong các phương thức thông dụng nhất của `String`, nó trả về độ dài chuỗi (Số ký tự của chuỗi).

Ví dụ:

```
string str="cntt";  
cout<<str.length()<<endl;
```

Kết quả → 4





## 4.3 Hàm xử lý trên chuỗi:

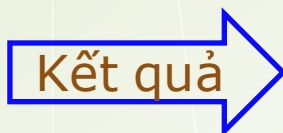
### 4.3.2 Xử lý string:

- Hàm append() (Nối chuỗi Appending):

Hàm append được thiết kế để nối thêm một chuỗi khác vào chuỗi gốc.

Ví dụ 1:

```
cout<<"\n Ham append : ";  
string str1="Khoa";  
string str2=" CNTT";  
cout<<str1.append(str2)<<endl;
```



Ham append : Khoa CNTT



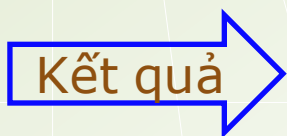
## 4.3 Hàm xử lý trên chuỗi:

### 4.3.2 Xử lý string:

- Hàm append() (Nối chuỗi Appending):

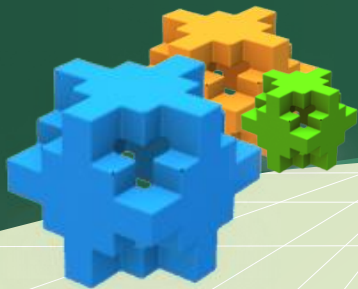
Ví dụ 2:

```
cout<<"\n Ham append : ";  
string str1="Khoa";  
string str2=" CNTT";  
cout<<str1.append(str2,0,3)<<endl;
```



Ham append : Khoa CN





## 4.3 Hàm xử lý trên chuỗi:

### 4.3.2 Xử lý string:

- Nối chuỗi (Appending):

Ví dụ 3:

```
cout<<"\n Ham append : ";  
string str1="Khoa CNTT";  
cout<<str1.append(3,'x')<<endl;
```

**Kết quả** → Khoa CNTTxxx



## 4.3 Hàm xử lý trên chuỗi:

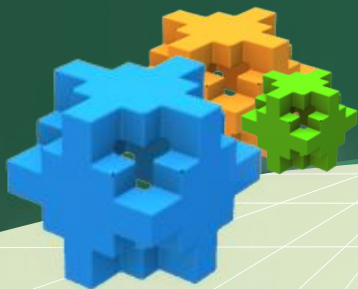
### 4.3.2 Xử lý string:

- Hàm `find()` là phương thức tìm vị trí xuất hiện của một chuỗi con trong chuỗi hiện tại. Phương thức này trả về hằng số `string::npos` nếu không tìm thấy.

Ví dụ 1:

```
cout<<"\n Ham find : ";  
string str1="Khoa CNTT";  
cout<<str1.find('C')<<endl;
```

**Kết quả** → Ham find : 5



## 4.3 Hàm xử lý trên chuỗi:

### 4.3.2 Xử lý string:

- Hàm `find()` là phương thức tìm vị trí xuất hiện của một chuỗi con trong chuỗi hiện tại. Phương thức này trả về hằng số `string::npos` nếu không tìm thấy.

Ví dụ 2:

```
cout<<"\n Ham append : ";  
string str1="Khoa CNTT!";  
cout<<str1.find('!',4)<<endl;
```

**Kết quả**

Ham find : 9



## 4.3 Hàm xử lý chuỗi:

### 4.3.2 Xử lý string:

- Hàm transform() (Chữ hoa-upper):

Lớp string không cung cấp một phương thức nào để chuyển đổi một chuỗi thành chuỗi chữ hoa hoặc chuỗi chữ thường, muốn chuyển chuỗi thành ký tự hoa sử dụng → thư viện transform trong C++.

Ví dụ 1:

```
string str="cntt";
```

```
std::transform(str.begin(),str.end(),str.begin(),::toupper);
```

```
cout<<str<<endl;
```

Kết quả → CNTT



## 4.3 Hàm xử lý trên chuỗi:

### 4.3.2 Xử lý string:

- Hàm transform() (Chữ thường- lower):

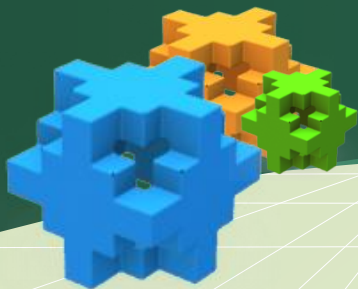
Muốn chuyển chuỗi thành ký tự thường sử dụng → thư viện transform trong C++.

Ví dụ 2:

```
string str="CNTT";  
std::transform(str.begin(),str.end(),str.begin(),::tolower);  
cout<<str<<endl;
```

Kết quả → cntt





## 4.3 Hàm xử lý trên chuỗi:

### 4.3.2 Xử lý string:

- Hàm `compare()` (So sánh chuỗi):

sử dụng những toán tử quan hệ (`==`, `!=`, `<`, `<=`, `>=`) được định nghĩa sẵn. Hoặc, sử dụng phương thức `compare()` so sánh một phần của một chuỗi

Ví dụ:

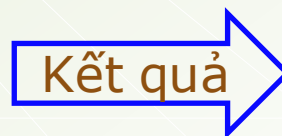
```
string str1="a";
```

```
string str2="A";
```

```
if(str1.compare(str2)==0)cout<<"\n str1 = str2";
```

```
else if(str1.compare(str2)<0) cout<<"\n str1 < str2";
```

```
else cout<<"\n str1 > str2";
```

Kết quả  `str1 > str2`





# Bài tập

- 1) Điền vào phần còn thiếu để tạo một biến greeting kiểu chuỗi và gán cho nó giá trị Hello.

?	?	=	?
---	---	---	---

- 1) Cho 2 chuỗi s1, s2. Hãy:

- Ghép chuỗi s2 vào s1.
- Tính độ dài của chuỗi s1

2. Đếm số từ trong chuỗi.

Ví dụ: nhập “công nghệ thông tin” → số từ trong chuỗi là 4

3. Kiểm tra một chuỗi có phải là chuỗi đối xứng không?

Chuỗi đối xứng là chuỗi : MADAM, ABCBA