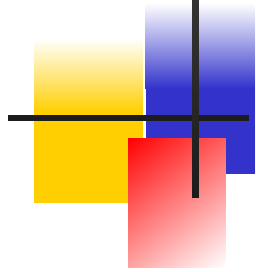




# Nhập và Xuất trong C

---

## Chương 4



# Mục tiêu của bài học

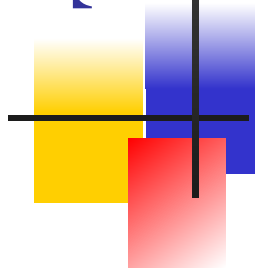
---

- Tìm hiểu các hàm định dạng Nhập/Xuất  
**scanf(), printf()**
- Sử dụng các hàm Nhập/Xuất ký tự  
**getchar(), putchar()**



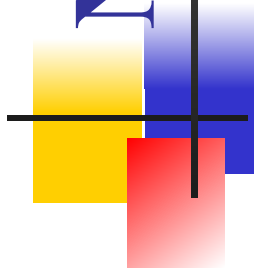
# Nhập/Xuất chuẩn

- Thư viện chuẩn trong C cung cấp các hàm xử lý cho việc nhập và xuất.
- Thư viện chuẩn có các hàm I/O, dùng để quản lý việc nhập, xuất, các thao tác trên ký tự và chuỗi.
- Thiết bị nhập chuẩn thường là bàn phím.
- Thiết bị xuất chuẩn thường là màn hình (console).
- Nhập và xuất có thể được xử lý qua các tập tin thay vì từ các thiết bị chuẩn.



# Tập tin Header `<stdio.h>`

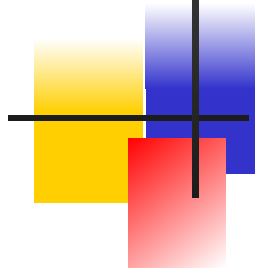
- `#include <stdio.h>`
- Đây là câu lệnh tiền xử lý
- `stdio.h` là tập tin header (header file)
- Chứa các macro sử dụng cho nhiều hàm nhập/xuất trong C
- Các macro trong `stdio.h` giúp các hàm `printf()`, `scanf()`, `putchar()`, `getchar()` thực thi



# Nhập/Xuất được định dạng

---

- `printf()` – Dùng cho xuất có định dạng
- `scanf()` – Sử dụng để nhập có định dạng
- *Các đặc tả định dạng* - qui định dạng thức mà theo đó giá trị của biến được nhập vào và in ra

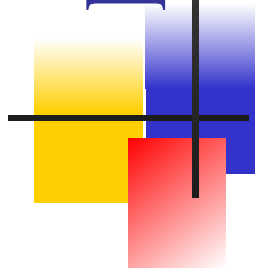


# printf ( )

- Được dùng để hiển thị dữ liệu ra thiết bị xuất chuẩn như màn hình (console)

Cú pháp → `printf ( "control string", argument list);`

- Danh sách đối số (argument list) chứa hằng, biến, biểu thức hoặc các hàm phân cách bởi dấu phẩy
- Phải có một lệnh định dạng trong “*control string*” cho mỗi đối số trong danh sách
- Các lệnh định dạng phải khớp với danh sách đối số về số lượng, kiểu và thứ tự.
- *control string* luôn được đặt trong dấu nháy kép “ ”, đây là dấu phân cách



# printf ( ) (tt.)

*control string* chứa một trong ba kiểu phần tử sau:

1. Các ký tự *văn bản* :  
gồm các ký tự có thể in được
2. Các *lệnh định dạng* :  
bắt đầu với ký hiệu % và theo sau là một mã định dạng tương ứng cho từng phần tử dữ liệu
3. Các *ký tự không in được* :  
gồm tab, blank và new\_line

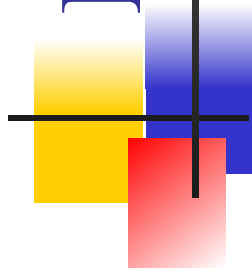


# Mã định dạng

Định dạng		printf()	scanf()
Ký tự đơn (single character)		%c	%c
Chuỗi (string)		%s	%s
Số nguyên có dấu (signed decimal integer)		%d	%d
Kiểu float - dạng dấu chấm thập phân (decimal notation)		%f	%f hoặc %e
Kiểu float - dạng dấu chấm thập phân		%lf	%lf
Kiểu float - dạng lũy thừa (exponential notation)		%e	%f or %e
Kiểu float ( %f hay %e , khi ngắn hơn)		%g	
Số nguyên không dấu (unsigned decimal integer)		%u	%u
Số nguyên hệ 16 không dấu - sử dụng “ABCDEF” (unsigned hexadecimal integer)		%x	%x
Số nguyên hệ 8 không dấu (unsigned octal integer)		%o	%o

**Trong bảng trên : c, d, f, lf, e, g, u, s, o và x là các bộ đặc tả kiểu**





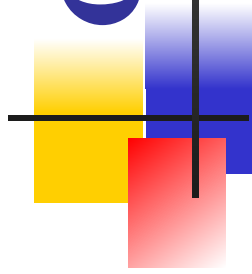
# Mã định dạng (tt.)

Mã định dạng	Các qui ước in
<code>%d</code>	Các con số trong số nguyên
<code>%f</code>	Các chữ số phần nguyên sẽ được in ra. Phần thập phân sẽ chỉ in 6 chữ số. Nếu phần thập phân ít hơn 6 chữ số, nó sẽ được thêm các chữ số 0 vào từ bên phải, ngược lại nó sẽ làm tròn số từ bên phải.
<code>%e</code>	Một con số bên trái của dấu chấm thập phân và 6 vị trí bên phải, như <code>%f</code> ở trên



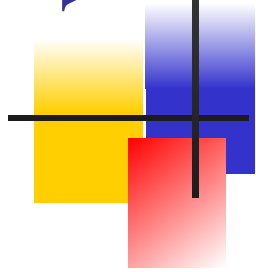
# Mã định dạng (tt.)

ST T	Lệnh	Chuỗi điều kiện	Nội dung chuỗi điều kiện	Danh sách đối số	Giải thích danh sách đối số	Hiển thị trên màn hình
1.	<code>printf(“%d”,300);</code>	<code>%d</code>	Chỉ chứa lệnh định dạng	300	Hằng	300
2.	<code>printf(“%d”,10+5);</code>	<code>%d</code>	Chỉ chứa lệnh định dạng	10 + 5	Biểu thức	15
3.	<code>printf(“Good Morning Mr. Lee.”);</code>	Good Morning Mr. Lee.	Chỉ chứa các ký tự văn bản	Rỗng	Rỗng	Good Morning Mr. Lee.
4.	<code>int count = 100; printf(“%d”,count);</code>	<code>%d</code>	Chỉ chứa lệnh định dạng	count	Biến	100
5.	<code>printf(“\nhello”);</code>	<code>\nhello</code>	Chứa ký tự không được in và các ký tự văn bản	Rỗng	Rỗng	hello on a new line
6.	<code>#define str “Good Apple “ ..... printf(“%s”,str);</code>	<code>%s</code>	Chỉ chứa lệnh định dạng	str	Hằng ký hiệu	Good Apple
7.	<code>..... int count,stud_num; count=0; stud_num=100; printf(“%d %d\n”,count,stud_num);</code>	<code>%d %d</code>	Chứa lệnh định dạng và ký tự không được in	count, stud_num	Hai biến	0 , 100



# Các ký tự đặc biệt

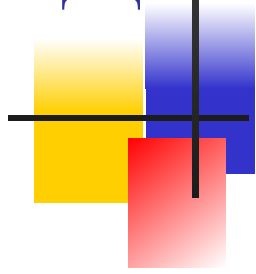
<code>\\</code>	In ra ký tự <code>\</code>
<code>\“</code>	In ra ký tự <code>“</code>
<code>%%</code>	In ra ký tự <code>%</code>



# Ví dụ cho hàm printf()

Chương trình hiển thị số nguyên, thập phân, ký tự và chuỗi

```
#include <stdio.h>
void main()
{
    int a = 10;
    float b = 24.67892345;
    char ch = 'A';
    printf("Integer data = %d", a);
    printf("Float Data = %f", b);
    printf("Character = %c", ch);
    printf("This prints the string");
    printf("%s", "This also prints a
    string");
}
```



# Bổ từ trong hàm printf()

## 1. Bổ từ ‘-‘

Phần tử dữ liệu sẽ được canh lề trái, phần tử sẽ được in bắt đầu từ vị trí bên trái trong cùng của trường.

## 2. Bổ từ xác định độ rộng trường

Có thể được sử dụng với kiểu float, double hoặc mảng ký tự (chuỗi). Độ rộng trường là một số nguyên xác định độ rộng nhỏ nhất cho phần tử dữ liệu.



# Bổ từ trong hàm printf() (tt.)

## 3. Độ chính xác

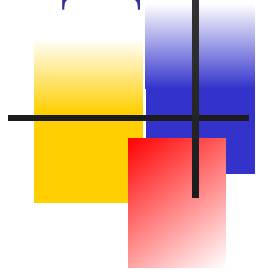
Được sử dụng với kiểu float, double hoặc mảng ký tự (chuỗi). Nếu dùng với kiểu float hay double, chuỗi con số xác định số lượng lớn nhất các con số được in bên phải dấu chấm thập phân.

## 4. Bổ từ '0'

Mặc định thì khoảng trống sẽ được thêm vào một trường. Nếu người dùng muốn thêm số 0 vào trường thì bổ từ '0' được dùng

## 5. Bổ từ 'l'

Bổ từ này có thể được dùng hiển thị các đối số nguyên kiểu int hay double. Mã định dạng tương ứng là %ld



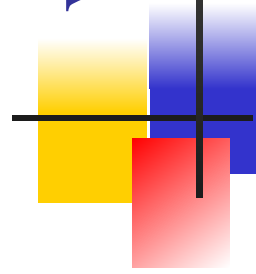
# Bổ từ trong hàm printf() (tt.)

## 6. Bổ từ 'h'

Bổ từ này được sử dụng để hiển thị dạng short int. Mã định dạng tương ứng như là %hd

## 7. Bổ từ '\*\*'

Nếu người dùng không muốn xác định độ rộng trường nhưng muốn chương trình xác định điều đó, bổ từ này được sử dụng



# Ví dụ về các bổ từ

*/\* This program demonstrate the use of Modifiers in printf() \*/*

```
#include <stdio.h>
void main() {
    printf("The number 555 in various forms:\n");
    printf("Without any modifier: \n");
    printf("[%d]\n", 555);
    printf("With - modifier : \n");
    printf("[%d]\n", 555);
    printf("With digit string 10 as modifier : \n");
    printf("[%10d]\n", 555);
    printf("With 0 as modifier : \n");
    printf("[%0d]\n", 555);
    printf("With 0 and digit string 10 as modifiers
:\n");
    printf("[%010d]\n", 555);
    printf("With -, 0 and digit string 10 as
modifiers:\n");
    printf("[% -010d]\n", 555);
}
```





# scanf()

---

- Được sử dụng để nhập dữ liệu  
Dạng tổng quát của hàm scanf()  
`scanf(“control string”, argument list);`
- Những định dạng dùng trong hàm  
`printf()` cũng được sử dụng với cùng cú  
pháp trong hàm `scanf()`



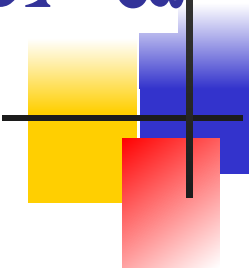
# Sự khác nhau về danh sách đối số giữa `printf()` và `scanf()`

---

- `printf()` sử dụng các tên biến, hằng, hằng biểu tượng và các biểu thức
- `scanf()` sử dụng các con trỏ tới biến

## Danh sách đối số trong `scanf()` phải theo qui tắc :

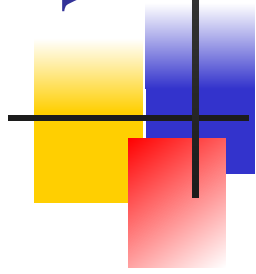
- Đọc giá trị vào một biến có kiểu dữ liệu cơ sở, sử dụng ký hiệu **&** trước tên biến
- Đọc giá trị vào một biến có kiểu dữ liệu dẫn xuất, không sử dụng **&** trước tên biến



# Sự khác nhau về các lệnh định dạng giữa `printf()` và `scanf()`

---

- Không có tùy chọn `%g`
- Mã định dạng `%f` và `%e` là giống nhau



# Ví dụ với hàm scanf()

```
#include <stdio.h>
void main() {
    int a;
    float d;
    char ch, name[40];
    printf("Please enter the data\n");
    scanf("%d %f %c %s", &a, &d, &ch, name);
    printf("\n The values accepted are:
           %d, %f, %c, %s", a, d, ch, name);
}
```



# Vùng đệm Nhập/Xuất

---

- Được sử dụng để đọc và viết các ký tự ASCII
- Một vùng đệm (buffer) là một không gian lưu trữ tạm thời trong bộ nhớ hoặc trên thẻ điều khiển thiết bị
- Bộ đệm Nhập/Xuất có thể chia làm :
  - Console I/O
  - Buffered File I/O



# Console I/O

---

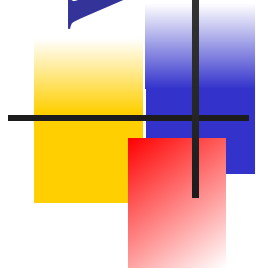
- Các hàm Console I/O chuyển các thao tác đến thiết bị xuất nhập chuẩn của hệ thống
- Trong ‘C’ các hàm **console I/O** đơn giản nhất là:
  - **getchar()** - đọc một và chỉ một ký tự từ bàn phím
  - **putchar()** - xuất một ký tự lên màn hình



# getchar()

---

- Dùng đọc dữ liệu nhập, một ký tự từ bàn phím
- Các ký tự đặt trong vùng đệm đến khi người dùng gõ phím enter
- Hàm `getchar()` không có đối số, nhưng vẫn phải có cặp dấu ngoặc `()`



# Ví dụ hàm getchar()

```
/*Program to demonstrate the use of getchar() */  
  
#include <stdio.h>  
void main()  
{  
    char letter;  
    printf("\nPlease enter any character:");  
    letter = getchar();  
    printf("\nThe character entered by you  
is %c", letter);  
}
```





# putchar()

---

- Hàm xuất ký tự trong ‘C’
- Có một đối số

**Đối số của một hàm putchar() có thể là :**

- Một hằng ký tự đơn
- Một mã định dạng
- Một biến ký tự



# Các tùy chọn

## và chức năng của putchar()

Đối số	Hàm	Chức năng
Biến ký tự	putchar(c)	Hiển thị nội dung của biến ký tự c
Hằng ký tự	putchar('A')	Hiển thị ký tự A
Hằng số	putchar('5')	Hiển thị số 5
Mã định dạng	putchar('\t')	Xen một khoảng trống tại vị trí con trỏ
Mã định dạng	putchar('\n')	Xen một lệnh xuống dòng tại vị trí con trỏ



# putchar()

**/\* This program demonstrates the use of  
constants and escape sequences in  
putchar() \*/**

```
#include <stdio.h>  
void main() {
```

```
    putchar('H'); putchar('\n');  
    putchar('\t');  
    putchar('E'); putchar('\n');
```

**Ví dụ**

```
    putchar('\t'); putchar('\t');  
    putchar('L'); putchar('\n');  
    putchar('\t'); putchar('\t'); putchar('\t');  
    putchar('L'); putchar('\n');  
    putchar('\t'); putchar('\t'); putchar('\t');  
    putchar('O');  
}
```