



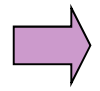
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI  
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

# TIN HỌC ĐẠI CƯƠNG

## Phần 3. Lập trình C

### Bài 7. Kiểu dữ liệu và biểu thức trong C

# Nội dung



7.1. Các kiểu dữ liệu chuẩn trong C

7.2. Khai báo và khởi tạo biến, hằng

7.3. Biểu thức trong C

7.4. Các phép toán trong C

7.5. Một số toán tử đặc trưng

7.6. Các lệnh vào ra dữ liệu với các biến

## 7.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
<b>unsigned char</b>	Kí tự	1 byte	$0 \div 255$
<b>char</b>	Kí tự	1 byte	$-128 \div 127$
<b>unsigned int</b>	Số nguyên không dấu	2 bytes	$0 \div 65,535$
<b>short int</b>	Số nguyên có dấu	2 bytes	$-32,768 \div 32,767$
<b>int</b>	Số nguyên có dấu	<b>2 bytes</b>	$-32,768 \div 32,767$

## 7.1. Các kiểu dữ liệu chuẩn trong C

Kiểu dữ liệu	Ý nghĩa	Kích thước	Miền dữ liệu
<b>unsigned long</b>	Số nguyên không dấu	4 bytes	$0 \div 4,294,967,295$
<b>long</b>	Số nguyên có dấu	4 bytes	$-2,147,483,648 \div 2,147,483,647$
<b>float</b>	Số thực dấu phẩy động, độ chính xác đơn	4 bytes	$\pm 3.4E-38 \div \pm 3.4E+38$
<b>double</b>	Số thực dấu phẩy động, độ chính xác kép	8 bytes	$\pm 1.7E-308 \div \pm 1.7E+308$
<b>long double</b>	Số thực dấu phẩy động, độ chính xác kép mở rộng	10 bytes	$\pm 3.4E-4932 \div \pm 1.1E+4932$

# Bảng mã ASCII

ASCII Code Chart

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

# Nội dung

7.1. Các kiểu dữ liệu chuẩn trong C



7.2. Khai báo và khởi tạo biến, hằng

7.3. Biểu thức trong C

7.4. Các phép toán trong C

7.5. Một số toán tử đặc trưng

7.6. Các lệnh vào ra dữ liệu với các biến

## 7.2.1. Khai báo và khởi tạo biến

- Một biến trước khi sử dụng phải được khai báo

- **Cú pháp:**

```
kieu_du_lieu ten_bien;
```

```
kieu_du_lieu ten_bien_1, ..., ten_bien_N;
```

- **Ví dụ:** Khai báo một biến x thuộc kiểu số nguyên 2 byte có dấu (int); biến y, z, t thuộc kiểu số thực 4 byte (float) như sau:

```
int x;
```

```
float y, z, t;
```

```
x = 3; y = x + 1;
```

## 7.2.1. Khai báo và khởi tạo biến (2)

### Kết hợp khai báo và khởi tạo

- *Cú pháp:*

```
kieu_du_lieu ten_bien = gia_tri_ban_dau;  
kieu_du_lieu bien_1 = gia_tri_1, bien_N =  
gia_tri_N;
```

- *Ví dụ:*

```
int a = 3; // sau lenh nay bien a se co  
gia tri bang 3
```

```
float x = 5.0, y = 7.6; // sau lenh nay x  
co gia tri 5.0, y co gia tri 7.6
```



## 7.2.2. Khai báo hằng

- Cách 1: Dùng từ khóa **#define**

- *Cú pháp:*

- #define** *ten\_hang* *gia\_tri*

- *Ví dụ:*

- `#define MAX_SINH_VIEN 50`

- `#define CNTT "Cong nghe thong tin"`

- `#define DIEM_CHUAN 23.5`

## 7.2.2. Khai báo hằng

- Cách 2: Dùng từ khóa **const** :

- *Cú pháp:*

- ```
const kieu_du_lieu ten_hang = gia_tri;
```

- *Ví dụ:*

- ```
const int MAX_SINH_VIEN = 50;
```

- ```
const char CNTT[20] = "Cong nghe thong tin";
```

- ```
const float DIEM_CHUAN = 23.5;
```

## 7.2.2. Khai báo hằng

- *Chú ý:*
  - Giá trị của các hằng phải được xác định ngay khi khai báo.
  - Trong chương trình, **KHÔNG thể thay đổi** được giá trị của hằng.
  - **#define** là chỉ thị tiền xử lý (preprocessing directive)
    - Dễ đọc, dễ thay đổi
    - Dễ chuyển đổi giữa các nền tảng phần cứng hơn
    - Tốc độ nhanh hơn

# Nội dung

7.1. Các kiểu dữ liệu chuẩn trong C

7.2. Khai báo và khởi tạo biến, hằng

➔ 7.3. Biểu thức trong C

7.4. Các phép toán trong C

7.5. Một số toán tử đặc trưng

7.6. Các lệnh vào ra dữ liệu với các biến

## 7.3.1. Các loại biểu thức

### a. Biểu thức số học:

- Là biểu thức mà giá trị của nó là các đại lượng số học (số nguyên, số thực).
- Các toán tử là các phép toán số học (cộng, trừ, nhân, chia...), các toán hạng là các đại lượng số học (số, biến, hằng).
- Ví dụ:

$$3 * 3.7$$

$$8 + 6 / 3$$

**a + b - c** // Với a, b, c là các biến thuộc một kiểu dữ liệu số nào đó

## 7.3.1. Các loại biểu thức

### b. Biểu thức logic:

- Là biểu thức mà giá trị của nó là các giá trị logic, tức là một trong hai giá trị: **ĐÚNG** (*TRUE*) hoặc **SAI** (*FALSE*).
  - Giá trị nguyên khác 0: **ĐÚNG** (*TRUE*),
  - Giá trị 0: **SAI** (*FALSE*).
- Các phép toán logic gồm có
  - AND: VÀ logic, kí hiệu là **&&**
  - OR: HOẶC logic, kí hiệu là **||**
  - NOT: PHỦ ĐỊNH, kí hiệu là **!**

## 7.3.1. Các loại biểu thức

- Ví dụ về biểu thức logic:

```
(5 > 7) && (9 != 10) // FALSE
0 || 1                // TRUE
(5 > 7) || (9 != 10) // TRUE
0                      // FALSE
!0                     // TRUE
3                      // TRUE
!3                     // FALSE
(a > b) && (a < b)    // FALSE
```

## 7.3.1. Các loại biểu thức

- c. Biểu thức quan hệ:
  - Là những biểu thức trong đó có sử dụng các toán tử quan hệ so sánh như lớn hơn, nhỏ hơn, bằng nhau, khác nhau ...
  - Chỉ có thể nhận giá trị là một trong 2 giá trị **ĐÚNG** (TRUE) hoặc **SAI** (FALSE)
  - Biểu thức quan hệ là một trường hợp riêng của biểu thức logic.



## 7.3.1. Các loại biểu thức

- Ví dụ về biểu thức quan hệ:

<code>5 &gt; 7</code>	<code>// FALSE</code>
<code>9 != 10</code>	<code>// TRUE</code>
<code>2 &gt;= 2</code>	<code>// TRUE</code>
<code>a &gt; b</code>	<code>// ???</code>
<code>a + 1 &gt; a</code>	<code>// TRUE</code>

# Sử dụng biểu thức

- Làm vế phải của lệnh gán.

$$x = (y + 1) / z$$

- Làm toán hạng trong các biểu thức khác
- Làm tham số thực trong lời gọi hàm
- Làm chỉ số trong các cấu trúc lặp **for**, **while**, **do while**.
- Làm biểu thức kiểm tra trong các cấu trúc rẽ nhánh **if**, **switch**.

# Nội dung

7.1. Các kiểu dữ liệu chuẩn trong C

7.2. Khai báo và khởi tạo biến, hằng

7.3. Biểu thức trong C

⇒ 7.4. Các phép toán trong C

7.5. Một số toán tử đặc trưng

7.6. Các lệnh vào ra dữ liệu với các biến

## Slide 19

---

**BL2**

**UPMF**

Ba-Vui Le, 10/17/2016

## 7.4. Các phép toán trong C

- Bao gồm:
  - Nhóm các phép toán số học
  - Nhóm các phép toán quan hệ
  - Nhóm các phép toán logic
  - Nhóm các phép toán thao tác trên bit
- Ngoài ra C còn cung cấp một số phép toán khác như phép gán, phép lấy địa chỉ ...

## 7.4.1. Phép toán số học

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
-	Phép đổi dấu	Số thực hoặc số nguyên	<code>int a, b;</code> <code>-12; -a; -25.6</code>
+	Phép toán cộng	Số thực hoặc số nguyên	<code>float x, y;</code> <code>5 + 8; a + x;</code> <code>3.6 + 2.9</code>
-	Phép toán trừ	Số thực hoặc số nguyên	<code>3 - 1.6; a - 5;</code>
*	Phép toán nhân	Số thực hoặc số nguyên	<code>a * b; b * y;</code> <code>2.6 * 1.7</code>
/	Phép toán chia	Số thực hoặc số nguyên	<code>10.0/3.0; (bằng 3.33)</code> <code>10/3.0; (bằng 3.33)</code> <code>10.0/3; (bằng 3.33)</code>
/	Phép chia lấy phần nguyên	Giữa 2 số nguyên	<code>10/3</code>
%	Phép chia lấy phần dư	Giữa 2 số nguyên	<code>10%3</code>

## 7.4.2. Phép toán trên bit

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
&	Phép VÀ nhị phân	2 số nhị phân	0 & 0 (có giá trị 0) 0 & 1 (có giá trị 0) 1 & 0 (có giá trị 0) 1 & 1 (có giá trị 1)  101 & 110 (có giá trị 100)
	Phép HOẶC nhị phân	2 số nhị phân	0   0 (có giá trị 0) 0   1 (có giá trị 1) 1   0 (có giá trị 1) 1   1 (có giá trị 1)  101   110 (có giá trị 111)

## 7.4.2. Phép toán trên bit

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Ví dụ
$\wedge$	Phép HOẶC LOẠI TRỪ 2 số nhị phân  TRỪ nhị phân	2 số nhị phân	$0 \wedge 0$ (có giá trị 0) $0 \wedge 1$ (có giá trị 1) $1 \wedge 0$ (có giá trị 1) $1 \wedge 1$ (có giá trị 0) $101 \wedge 110$ (có giá trị 011)
$\ll$	Phép DỊCH TRÁI nhị phân	Số nhị phân	$a \ll n$ (có giá trị $a \cdot 2^n$ ) $101 \ll 2$ (có giá trị 10100)
$\gg$	Phép DỊCH PHẢI nhị phân	Số nhị phân	$a \gg n$ (có giá trị $a / 2^n$ ) $101 \gg 2$ (có giá trị 1)
$\sim$	Phép ĐẢO BIT Phép lấy BÙ 1	Số nhị phân	$\sim 0$ (có giá trị 1) $\sim 1$ (có giá trị 0) $\sim 110$ (có giá trị 001)



## 7.4.3. Phép toán quan hệ

Toán tử	Ý nghĩa	Ví dụ
$>$	So sánh lớn hơn giữa 2 số nguyên hoặc thực.	$2 > 3$ (có giá trị 0) $6 > 4$ (có giá trị 1) $a > b$
$>=$	So sánh lớn hơn hoặc bằng giữa 2 số nguyên hoặc thực.	$6 >= 4$ (có giá trị 1) $x >= a$
$<$	So sánh nhỏ hơn giữa 2 số nguyên hoặc thực.	$5 < 3$ (có giá trị 0),
$<=$	So sánh nhỏ hơn hoặc bằng giữa 2 số nguyên hoặc thực.	$5 <= 5$ (có giá trị 1) $2 <= 9$ (có giá trị 1)
$==$	So sánh bằng nhau giữa 2 số nguyên hoặc thực.	$3 == 4$ (có giá trị 0) $a == b$
$!=$	So sánh không bằng (so sánh khác) giữa 2 số nguyên hoặc thực.	$5 != 6$ (có giá trị 1) $6 != 6$ (có giá trị 0)

## 7.4.4. Phép toán logic

Toán tử	Ý nghĩa	Kiểu dữ liệu của toán hạng	Vi dụ
&&	Phép VÀ LOGIC. Biểu thức VÀ LOGIC bằng 1 khi và chỉ khi cả 2 toán hạng đều bằng 1	Hai biểu thức logic	$3 < 5 \ \&\& \ 4 < 6$ (có giá trị 1) $2 < 1 \ \&\& \ 2 < 3$ (có giá trị 0) $a > b \ \&\& \ c < d$
	Phép HOẶC LOGIC. Biểu thức HOẶC LOGIC bằng 0 khi và chỉ khi cả 2 toán hạng bằng 0.	Hai biểu thức logic	$6 \    \ 0$ (có giá trị 1) $3 < 2 \    \ 3 < 3$ (có giá trị 0) $x \geq a \    \ x == 0$
!	Phép PHỦ ĐỊNH LOGIC một ngôi. Biểu thức PHỦ ĐỊNH LOGIC có giá trị bằng 1 nếu toán hạng bằng 0 và có giá trị bằng 0 nếu toán hạng bằng 1	Biểu thức logic	$!3$ (có giá trị 0) $!(2 > 5)$ (có giá trị 1)

## 7.4.5. Phép toán gán

- Cú pháp:

**tên\_biến = biểu\_thức;**

- Lấy giá trị của *biểu\_thức* gán cho *tên\_biến*
- Ví dụ:

```
int a, b, c;
```

```
a = 3;
```

```
b = a + 5;
```

```
c = a * b;
```

## 7.4.5. Phép toán gán

- Biểu thức gán là biểu thức nên nó cũng có giá trị.
- Giá trị của biểu thức gán bằng giá trị của biểu\_thức:  
→ Có thể gán giá trị của biểu thức gán cho một biến khác hoặc sử dụng như một biểu thức bình thường
- Ví dụ:

```
int a, b, c;
```

```
a = b = 2007;
```

```
c = (a = 20) * (b = 30); // c = ?
```

## 7.4.5. Phép toán gán

- Phép toán gán thu gọn:  
 $\mathbf{x = x + y;}$  giống như  
 $\mathbf{x += y;}$
- Dạng lệnh gán thu gọn này còn áp dụng được với các phép toán khác:  $+$ ,  $-$ ,  $*$ ,  $/$ ,  $\%$ ,  $>>$ ,  $<<$ ,  $\&$ ,  $|$ ,  $^$

## 7.4.6. Thứ tự ưu tiên các phép toán

Mức	Các toán tử	Trật tự kết hợp
1	() [] . -> ++(hậu tố) --(hậu tố)	----->
2	! ~ ++(tiền tố) --(tiền tố) - * & sizeof	<-----
3	* / %	----->
4	+ -	----->
5	<< >>	----->
6	< <= > >=	----->
7	== !=	----->
8	&	----->
9	^	----->
10		----->
11	&&	----->
12		----->
13	? :	<-----
14	= += -=	<-----

# Ví dụ

Ví dụ 1:

`a < 10 && 2 * b < c`

`→ ( a < 10 ) && ( ( 2 * b ) < c )`

Ví dụ 2:

`int a = 35, b = 14, c = 5, d = 6;`

`int e;`

`e = (a + b) * c / d;`

`e = ((a + b) * c) / d;`

`e = (a + b) * (c / d);`

`e = a + (b * c) / d;`

# Nội dung

7.1. Các kiểu dữ liệu chuẩn trong C

7.2. Khai báo và khởi tạo biến, hằng

7.3. Biểu thức trong C

7.4. Các phép toán trong C



7.5. Một số toán tử đặc trưng

7.6. Các lệnh vào ra dữ liệu với các biến



## 7.5.1. Các phép toán tăng giảm một đơn vị

- Tăng hoặc giảm một đơn vị cho biến:

`<tên biến> = <tên biến> + 1;`

`→ <tên biến>++;`

`<tên biến> = <tên biến> - 1;`

`→ <tên biến>--;`

**Ví dụ:**

```
int a = 5;
```

```
float x = 10;
```

```
a++; // tương đương với a = a + 1;
```

```
x--; // tương đương với x = x - 1;
```

# Tiền tố và hậu tố

- **Tiền tố:** Thay đổi giá trị của biến trước khi sử dụng
- **Hậu tố:** Tính toán giá trị của biểu thức bằng giá trị ban đầu của biến, sau đó mới thay đổi giá trị của biến
- Ví dụ:

```
int a, b, c;  
a = 3;           // a bằng 3  
b = a++;         // Dạng hậu tố  
                 // b bằng 3, a bằng 4  
c = ++b;         // Dạng tiền tố  
                 // b bằng 4, c bằng 4
```

## 7.5.2. Phép toán lấy địa chỉ biến (&)

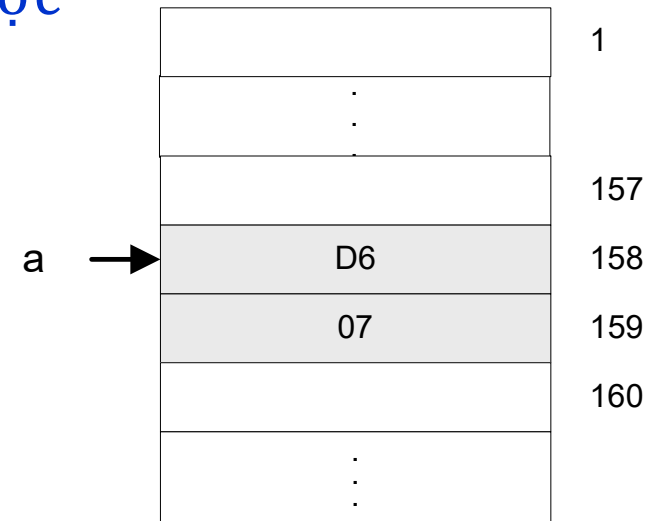
- Biến thực chất là một vùng nhớ được đặt tên (là tên của biến) trên bộ nhớ của máy tính.
- Mọi ô nhớ trên bộ nhớ máy tính đều được đánh địa chỉ. Do đó mọi biến đều có địa chỉ.
- Cú pháp:

**&<tên biến>;**

- Ví dụ:

```
int a = 2006;
```

→ `&a;` // có giá trị là 158 hay 9E



### 7.5.3. Phép toán chuyển đổi kiểu bắt buộc

- Chương trình dịch sẽ tự động chuyển đổi kiểu  
`char → int → long int → float → double  
→ long double`
- Ngược lại
  - Số nguyên **long int** 50,000 không phải là một số nguyên kiểu **int** vì phạm vi biểu diễn của kiểu **int** là từ (-32,768 đến 32,767).  
→ Phải ép kiểu
- Cú pháp:  
`(<kiểu dữ liệu mới>) <biểu thức>;`

## 7.5.3. Phép toán chuyển đổi kiểu bắt buộc (2)

- Ví dụ:

```
#include <stdio.h>
#include <conio.h>
void main()
{
    long int li; int i; float f;
    clrscr();
    li = 0x123456; f = 123.456;
    i = (int) li;
    printf("\n li = %ld; i = %d", li, i);
    i = (int) f;
    printf("\n f = %f; i = %d", f, i);
    getch();
}
```

### Kết quả

li = 1193046; i = 13398

f = 123.456000; i = 123

## 7.5.4. Biểu thức điều kiện

- Cú pháp

**biểu\_thức\_1 ? biểu\_thức\_2 : biểu\_thức\_3**

Giá trị của biểu thức điều kiện

- Giá trị của biểu\_thức\_2 nếu biểu\_thức\_1 có giá trị khác 0 (tương ứng với giá trị logic ĐÚNG),
- Ngược lại: Giá trị của biểu\_thức\_3 nếu biểu\_thức\_1 có giá trị bằng 0 (tương ứng với giá trị logic SAI).

- Ví dụ:

```
float x, y, z;      // khai báo biến
x = 3.8; y = 7.6;    // gán giá trị cho các biến x, y
z = (x < y) ? x : y; // z sẽ có giá trị bằng giá trị
                    // nhỏ nhất trong 2 số x và y
```

# Nội dung

7.1. Các kiểu dữ liệu chuẩn trong C

7.2. Khai báo và khởi tạo biến, hằng

7.3. Biểu thức trong C

7.4. Các phép toán trong C

7.5. Một số toán tử đặc trưng

⇒ 7.6. Các lệnh vào ra dữ liệu





## 7.6. Các lệnh vào ra dữ liệu

- C cung cấp 2 hàm vào ra cơ bản:
  - `printf()`
  - `scanf()`
- Muốn sử dụng 2 hàm `printf()` và `scanf()` ta cần khai báo tệp tiêu đề `stdio.h`:  
`#include <stdio.h>`  
hoặc  
`#include "stdio.h"`

## 7.6.1. Hàm printf()

### a. Mục đích và cú pháp:

- Mục đích:
  - Hiển thị ra màn hình các loại dữ liệu cơ bản như: Số, kí tự và chuỗi kí tự
  - Định dạng dữ liệu được hiển thị
  - Một số hiệu ứng hiển thị đặc biệt như xuống dòng, sang trang,...

## a. Mục đích và cú pháp (2)

- Cú pháp:

```
printf(xau_dinh_dang [, danh_sach_tham_so] );
```

- **xau\_dinh\_dang**: Qui định cách thức hiển thị dữ liệu ra màn hình máy tính.
- **danh\_sach\_tham\_so**: Danh sách các biến sẽ được hiển thị giá trị lên màn hình theo cách thức được qui định trong **xau\_dinh\_dang**.

## a. Mục đích và cú pháp (3)

- Ví dụ:

```
#include <conio.h>
#include <stdio.h>
void main()
{   int a = 5;
    float x = 1.234;
    printf("Hien thi mot so nguyen %d và mot
so thuc %f", a, x);
    getch();
}
```

- Sẽ cho ra kết quả:

Hien thi mot so nguyen 5 va mot so thuc 1.234000

## a. Mục đích và cú pháp (4)

- Trong **xau\_dinh\_dang** chứa:
  - Các kí tự thông thường: Được hiển thị ra màn hình.
  - Các nhóm kí tự định dạng: Xác định quy cách hiển thị các tham số trong phần **danh\_sach\_tham\_so**.
  - Các kí tự điều khiển: Dùng để tạo các hiệu ứng hiển thị đặc biệt như xuống dòng (“\n”) hay sang trang (“\f”)...

```
printf(xau_dinh_dang [, danh_sach_tham_so]);
```

## a. Mục đích và cú pháp (5)

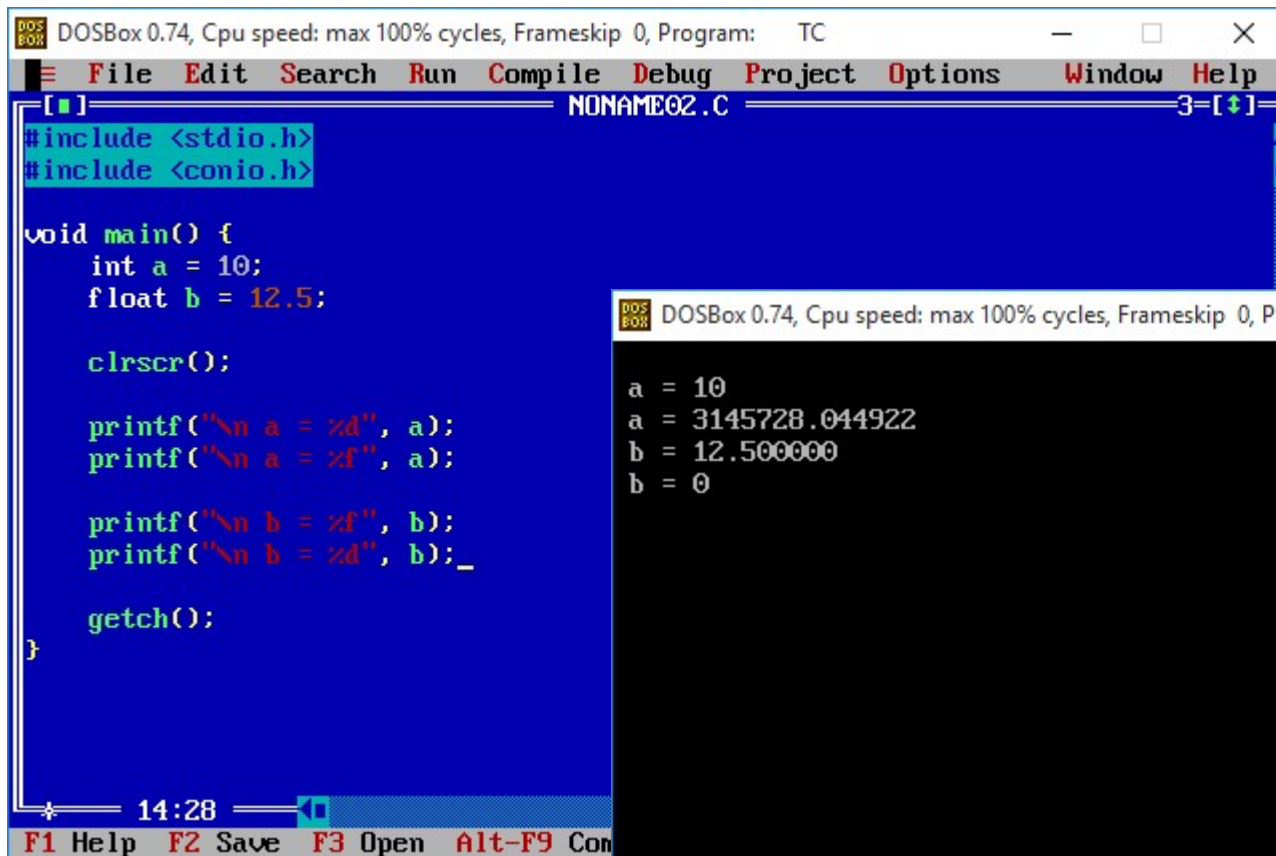
- Mỗi nhóm kí tự định dạng chỉ dùng cho một kiểu dữ liệu

Ví dụ: %d dùng cho kiểu nguyên

%f dùng cho kiểu thực

- Nếu giữa nhóm kí tự định dạng và tham số tương ứng không phù hợp với nhau thì sẽ hiển thị ra kết quả không như ý.

# Ví dụ



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

File Edit Search Run Compile Debug Project Options Window Help

NONAME02.C 3=[↑]

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

void main() {
    int a = 10;
    float b = 12.5;

    clrscr();

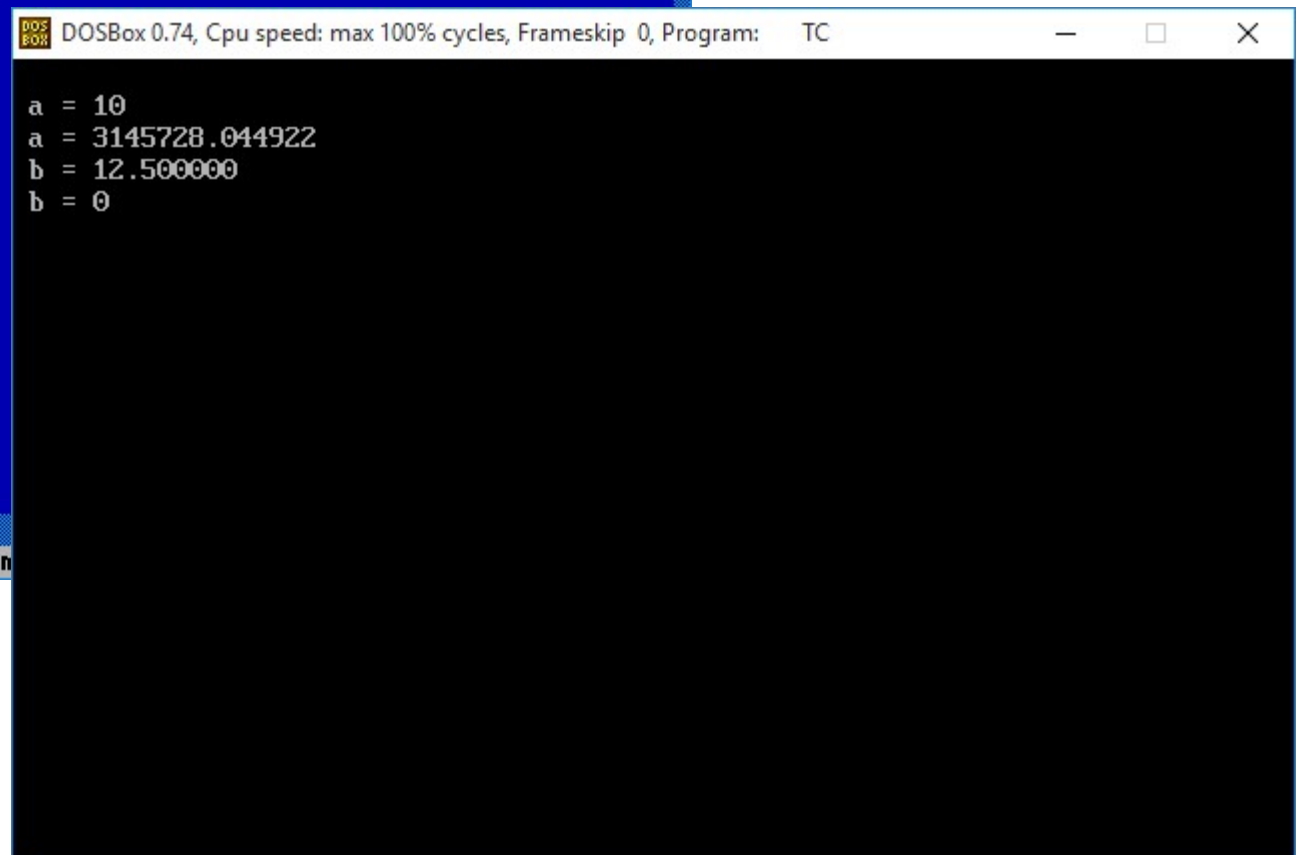
    printf("\n a = %d", a);
    printf("\n a = %f", a);

    printf("\n b = %f", b);
    printf("\n b = %d", b);_

    getch();
}
```

14:28

F1 Help F2 Save F3 Open Alt-F9 Con



DOSBox 0.74, Cpu speed: max 100% cycles, Frameskip 0, Program: TC

```
a = 10
a = 3145728.044922
b = 12.500000
b = 0
```

## a. Mục đích và cú pháp (6)

- **danh\_sach\_tham\_so** phải phù hợp với các nhóm kí tự định dạng trong **xau\_dinh\_dang** về:
  - Số lượng
  - Kiểu dữ liệu
  - Thứ tự



## b. Một số nhóm định dạng phổ biến

Nhóm kí tự định dạng	Kiểu dữ liệu	Kết quả
<b>%c</b>	<b>int, char</b>	<b>Kí tự đơn lẻ</b>
<b>%i, %d</b>	<b>int, char</b>	<b>Số thập phân</b>
<b>%o</b>	<b>int, char</b>	<b>Số bát phân</b> (không có 0 đằng trước)
<b>%x, %X</b>	<b>int, char</b>	<b>Số hexa</b> (chữ thường/chữ hoa)
<b>%u</b>	<b>unsigned int/char</b>	<b>Số thập phân</b>

## b. Một số nhóm định dạng phổ biến (2)

Nhóm kí tự định dạng	Kiểu dữ liệu	Kết quả
%ld, %li	long	Số thập phân
%lo	long	Số bát phân (không có 0 đằng trước)
%lx, %LX	long	Số hexa (chữ thường/chữ hoa)
%lu	unsigned long	Số thập phân

## b. Một số nhóm định dạng phổ biến (3)

Nhóm kí tự định dạng	Kiểu dữ liệu	Kết quả
%s	char []	Hiển thị chuỗi kí tự kết thúc bởi '\0'
%f	float/double	Số thực dấu phẩy tĩnh
%e, %E	float/double	Số thực dấu phẩy động

## c. Độ rộng hiển thị - số nguyên

- Đối với **số nguyên** hoặc **ký tự** hoặc **xâu ký tự**:
  - Có dạng `%md`, với **m** là số nguyên không âm
  - Ví dụ: Có số `a = 1234`

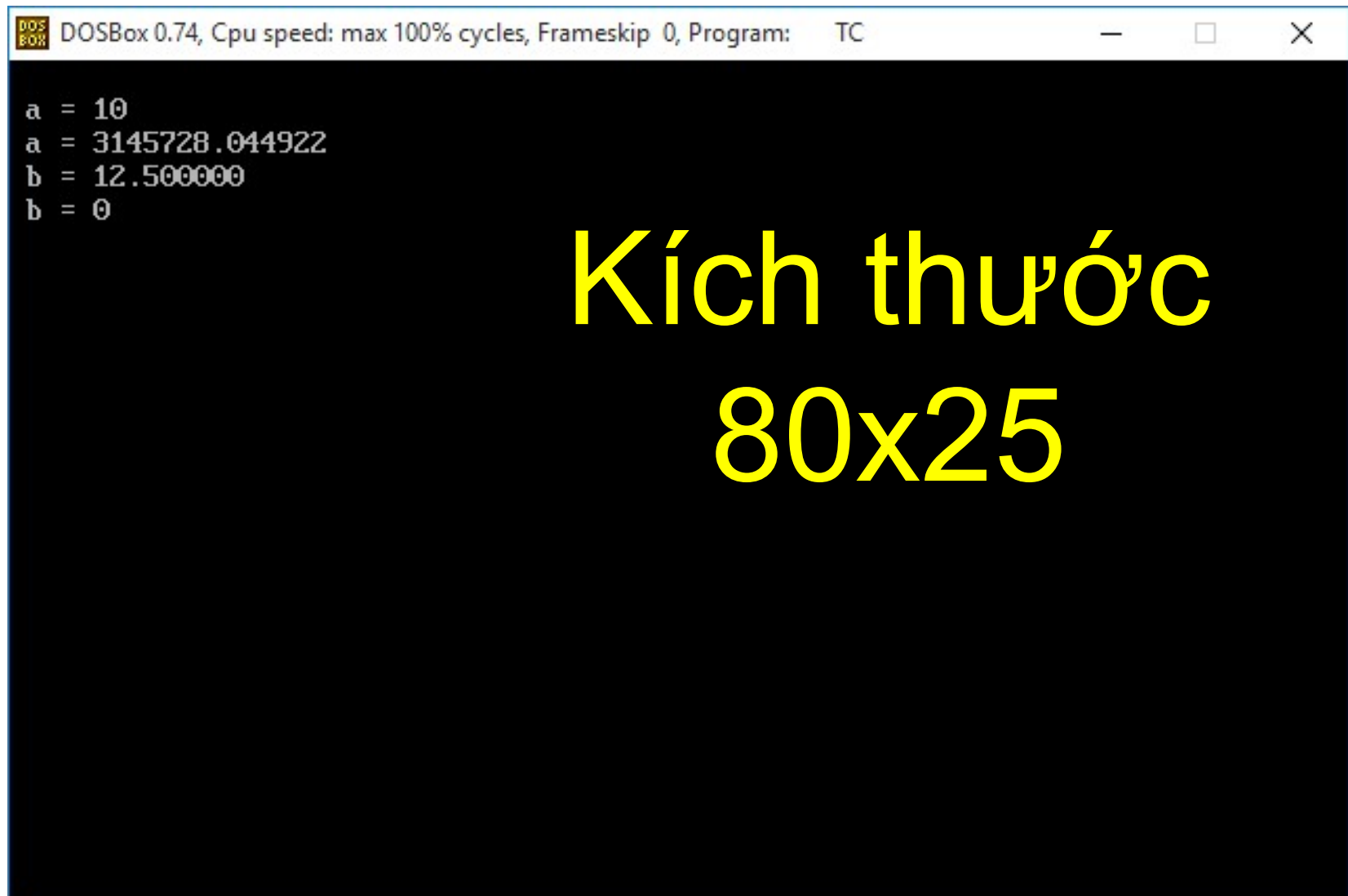
Lệnh:

```
printf("%5d", a); //đanh 5 cho de hien thi a  
printf("\n%5d", 34);
```

Cho ra kết quả: □1234  
                  □□□34

(□ kí hiệu cho dấu cách đơn (*space*) )

# Màn hình hiển thị MS-DOS



## c. Độ rộng hiển thị - số nguyên (2)

- Ví dụ:

```
printf("\n%3d %15s %3c", 1, "nguyen van a", 'g');  
printf("\n%3d %15s %3c", 2, "tran van b", 'k');
```

- Kết quả:

```
  1      nguyen van a  g  
  2      tran van b  k
```

## c. Độ rộng hiển thị - số thực

- m, n là 2 số nguyên không âm

**%m.nf**

Trong đó:

- m vị trí để hiển thị số thực, bao gồm cả kí tự thập phân '.', và ký tự đảo dấu '-'
- n vị trí trong m vị trí đó để hiển thị phần thập phân.

## c. Độ rộng hiển thị - số thực (2)

- Ví dụ:

```
printf("\n%f", 17.345);  
printf("\n%.2f", 17.345);  
printf("\n%7.2f", 17.345);
```

- Kết quả:

17.345000

17.35

□□17.35



## c. Độ rộng hiển thị - Chú ý

- Khi số chỗ cần thiết để hiển thị nội dung dữ liệu lớn hơn trong định dạng:
  - Tự động cung cấp thêm chỗ mới để hiển thị chứ không cắt bớt nội dung của dữ liệu.

– Ví dụ:

```
int a = 1000;  
printf("So a la: %1d", a);
```

– Kết quả:

```
So a la: 1000
```

## d. Căn lề phải, lề trái

- **Căn lề phải:**
  - Khi hiển thị dữ liệu, mặc định C căn lề phải
- **Căn lề trái:**
  - Nếu muốn căn lề trái khi hiển thị dữ liệu ta chỉ cần thêm dấu trừ - vào ngay sau dấu %.

## d. Căn lề phải, lề trái (2)

- Ví dụ:


```
printf("\n%-3d %-15s %.2f %-3c", 9, "nguyen  
van a", 7.5, 'g');
```

```
printf("\n%-3d %-15s %.2f %-3c", 10, "nguyen  
ha", 6.75, 'k');
```

- Kết quả:

9    nguyen van a    7.50 g

10   nguyen ha    6.75 k



The screenshot shows a C++ IDE with a source code editor and an output window. The source code is as follows:

```
LE_TRAI.CPP 1
#include <stdio.h>
#include <conio.h>
void main()
{
    printf("\n%-3d %-15s %.2f %-3c", 9, "nguyen van a", 7.5, 'g');
    printf("\n%-3d %-15s %.2f %-3c", 10, "nguyen ha", 6.75, 'k');
}
1:1
```

The output window shows the following results:

```
Output 2-[↑]
C:\TC\BIN>tc
9   nguyen van a   7.50 g
10  nguyen ha     6.75 k
```

## 7.6.2. Hàm scanf()



### a. Mục đích và cú pháp:

- Mục đích:
  - Hàm **scanf()** dùng để nhập dữ liệu từ bàn phím
- Cú pháp:  
`scanf(xau_dinh_dang [, danh_sach_dia_chi]);`
- Ví dụ:  

```
int a; float b;  
scanf("%d%f", &a, &b);
```

## a. Mục đích và cú pháp (3)



- **xâu\_dinh\_dang:**
  - Gồm các ký tự được quy định cho từng loại dữ liệu được nhập vào.
  - Ví dụ: với dữ liệu định nhập vào là kiểu nguyên thì xâu định dạng là : %d
- **danh\_sach\_dia\_chi:**
  - Bao gồm các địa chỉ của các biến (toán tử &), phân tách nhau bởi dấu phẩy (,)

## a. Mục đích và cú pháp (4)



- **danh\_sach\_dia\_chi** phải phù hợp với các nhóm kí tự định dạng trong **xau\_dinh\_dang** về:
  - Số lượng biến cần nhập
  - Kiểu dữ liệu
  - Thứ tự

## b. Một số nhóm định dạng phổ biến

Nhóm kí tự định dạng	Kiểu dữ liệu	Chú thích	Ví dụ
%c	char	Kí tự đơn lẻ	char x; scanf("%c", &x);
%d	int	Số thập phân	int x; scanf("%d", &x);
%o	int	Số bát phân	int x; scanf("%o", &x);
%x	int	Số hexa	int x; scanf("%x", &x);
%u	unsigned int	Số thập phân	unsigned int x; scanf("%u", &x);

## c. Một số nhóm định dạng phổ biến (3)

Nhóm kí tự định dạng	Kiểu dữ liệu	Chú thích
%s	char[]	Nhập chuỗi kí tự kết thúc bởi '\0'
%f	float	Số thực dấu phẩy thập
%ld	long	Số nguyên
%lf	double	Số thực dấu phẩy thập



# Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main()
{
    // khai bao bien
    int a; float x;
    char ch; char str[30];
    // Nhap du lieu
    printf("Nhap vao mot so nguyen");
    scanf("%d", &a);
    printf("\n Nhap vao mot so thuc");
    scanf("%f", &x);
```

## Ví dụ

```
printf("\n Nhap vao mot ki tu");  
fflush(stdin); scanf("%c", &ch);  
printf("\n Nhap vao mot xau ki tu");  
fflush(stdin); scanf("%s", str);  
  
// Hien thi du lieu vua nhap vao  
printf("\n Nhung du lieu vua nhap vao");  
printf("\n So nguyen: %d", a);  
printf("\n So thuc : %.2f", x);  
printf("\n Ki tu: %c: ", ch);  
printf("\n Xau ki tu: %s", str);  
getch();  
}
```

# Ví dụ



- Kết quả:

Nhap vao mot so nguyen: 2007

Nhap vao mot so thuc: 17.1625

Nhap vao mot ki tu: b

Nhap vao mot xau ki tu: ngon ngu lap  
trinh C

Nhung du lieu vua nhap vao

So nguyen: 2007

So thuc: 17.16

Ki tu: b

Xau ki tu: ngon

# Ví dụ

```
SCANF.CPP 1=111
void main()
{
    // khai bao bien
    int a; float x;
    char ch; char str[30];
    // Nhap du lieu
    printf("Nhap vao mot so nguyen: ");
    scanf("%d",&a);
    printf("\n Nhap vao mot so thuc: ");
    scanf("%f",&x);
    printf("\n Nhap vao mot ki tu: ");
    fflush(stdin); scanf("%c",&ch);
    printf("\n Nhap vao mot xau ki tu: ");
    fflush(stdin); scanf("%s",str);

    // Hien thi du lieu vua nhap vao
    printf("\n Nhung du lieu vua nhap vao");
    printf("\n So nguyen: %d",a);
    printf("\n So thuc : %.2f",x);
    printf("\n Ki tu: %c: ",ch);
    printf("\n Xau ki tu: %s",str);
}
4:2
```

```
Output 2
Nhap vao mot so nguyen: 2007

Nhap vao mot so thuc: 17.1625

Nhap vao mot ki tu: b

Nhap vao mot xau ki tu: ngon ngu

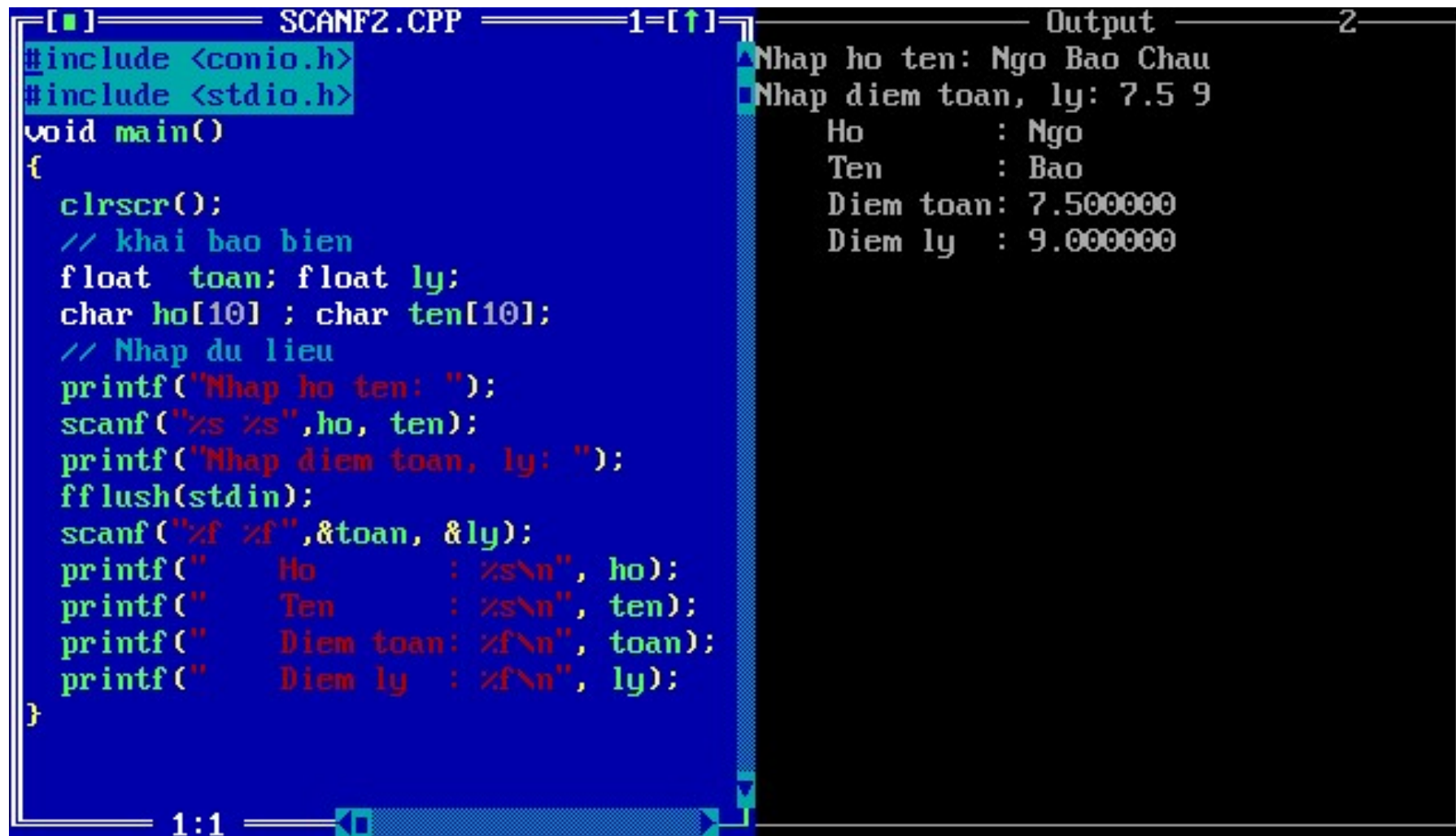
Nhung du lieu vua nhap vao
So nguyen: 2007
So thuc : 17.16
Ki tu: b:
Xau ki tu: ngon
```

## c. Một số quy tắc cần lưu ý



- Quy tắc 1: Khi đọc số (%i, %d, %f)
  - Hàm **scanf()** quan niệm rằng mọi kí tự số, dấu chấm (".") đều là kí tự hợp lệ.
  - Khi gặp các dấu phân cách như tab, xuống dòng hay dấu cách (space bar) thì **scanf()** sẽ hiểu là kết thúc nhập dữ liệu cho một số

## c. Một số quy tắc cần lưu ý (tiếp)



```
[■]===== SCANF2.CPP =====1=[↑]
#include <conio.h>
#include <stdio.h>
void main()
{
    clrscr();
    // khai bao bien
    float toan; float ly;
    char ho[10]; char ten[10];
    // Nhap du lieu
    printf("Nhap ho ten: ");
    scanf("%s %s", ho, ten);
    printf("Nhap diem toan, ly: ");
    fflush(stdin);
    scanf("%f %f", &toan, &ly);
    printf("    Ho      : %s\n", ho);
    printf("    Ten     : %s\n", ten);
    printf("    Diem toan: %f\n", toan);
    printf("    Diem ly  : %f\n", ly);
}

1:1
```

Output 2

```
Nhap ho ten: Ngo Bao Chau
Nhap diem toan, ly: 7.5 9
Ho      : Ngo
Ten     : Bao
Diem toan: 7.500000
Diem ly  : 9.000000
```

## c. Một số quy tắc cần lưu ý (tiếp)



- Quy tắc 2: Khi đọc kí tự (%c):

Hàm **scanf()** cho rằng mọi kí tự có trong bộ đệm của thiết bị vào chuẩn đều là hợp lệ, kể cả các kí tự tab, xuống dòng hay dấu cách.



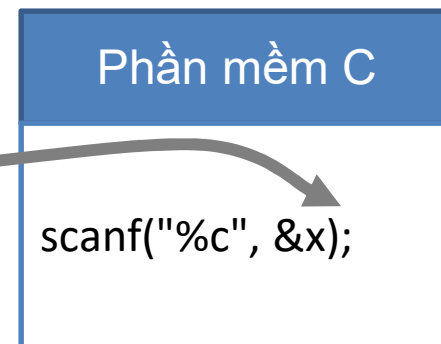
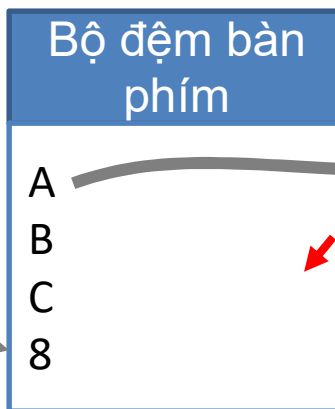
Phần mềm  
(viết bằng C)

## c. Một số quy tắc cần lưu ý (tiếp)



- Quy tắc 3: Khi đọc chuỗi ký tự (%s):
  - + Hàm **scanf()** nếu gặp các ký tự dấu trắng, dấu tab hay dấu xuống dòng thì nó sẽ hiểu là kết thúc nhập dữ liệu cho một chuỗi ký tự.
  - + Trước khi nhập dữ liệu ta nên dùng lệnh **fflush(stdin)** để xóa bộ đệm.

người dùng  
gõ ký tự '8'





## 7.6.3. Các lệnh vào ra khác

- **Hàm `gets()` :**

Dùng để nhập vào từ bàn phím một chuỗi kí tự **bao gồm cả dấu cách**, điều mà hàm `scanf()` không làm được.

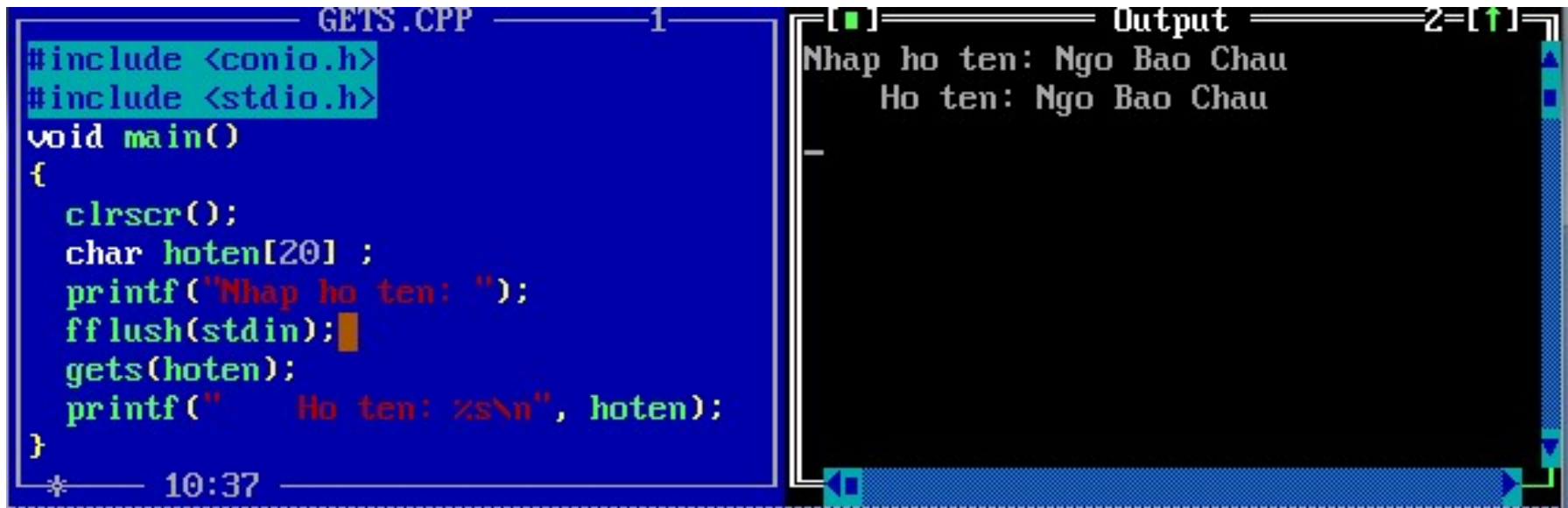
- Cú pháp :

`gets(biến_xâu_kí_tự) ;`

- Ví dụ:

```
char str[30];  
printf("Nhap vao mot xau ki tu:");  
fflush(stdin); gets(str);
```

## 7.6.3. Các lệnh vào ra khác



```
GETS.CPP 1
#include <conio.h>
#include <stdio.h>
void main()
{
    clrscr();
    char hoten[20] ;
    printf("Nhap ho ten: ");
    fflush(stdin);
    gets(hoten);
    printf("    Ho ten: %s\n", hoten);
}
* 10:37
```

Output

Nhap ho ten: Ngo Bao Chau  
Ho ten: Ngo Bao Chau

## 7.6.3. Các lệnh vào ra khác (2)

- **Hàm puts():**

Hiển thị ra màn hình nội dung **xâu\_kí\_tự** và sau đó đưa con trỏ xuống dòng mới.

- Cú pháp:

**puts (xâu\_kí\_tự) ;**

- Ví dụ:

**puts ("Nhập vào xâu kí tự:");**

- Tương đương với:

**printf ("%s\n", "Nhập vào xâu kí tự:");**

**Hoặc**

**printf ("Nhập vào xâu kí tự:\n");**

### 7.6.3. Các lệnh vào ra khác (3)

- Hàm `getch()`: thường dùng để chờ người sử dụng ấn một phím bất kì rồi sẽ kết thúc chương trình.
- Cú pháp

**`getch () ;`**

- Sử dụng hàm **`gets ()`** , **`puts ()`** , cần khai báo tệp tiêu đề **`stdio.h`**
- Sử dụng hàm **`getch ()`** , cần khai báo tệp tiêu đề **`conio.h`**

# Ví dụ

```
#include <conio.h>
#include <stdio.h>
void main()
{
    char str[30];
    puts("Hay cho biet ho ten ban:");
    fflush(stdin); gets(str);
    printf("Xin chao ");
    puts(str);
    puts("An phim bat ki de ket thuc...");
    getch();
}
```

# Ví dụ

- Kết quả:

Hay cho biet ho ten ban:

ngon ngu lap trinh C

Xin chao ngon ngu lap trinh C

An phim bat ki de ket thuc ...



The screenshot displays a code editor window titled 'PUTS.CPP' on the left and an 'Output' window on the right. The source code in the editor is as follows:

```
#include <conio.h>
#include <stdio.h>
void main()
{
    char str[30]; clrscr();
    puts("Hay cho biet ho ten ban: ");
    fflush(stdin); gets(str);
    printf("Xin chao ");
    puts(str);
    puts("An phim bat ki de ket thuc...");
    getch();
}
```

The output window shows the program's execution results:

```
Hay cho biet ho ten ban:
Ngo Bao Chau
Xin chao Ngo Bao Chau
An phim bat ki de ket thuc...
```

The status bar at the bottom of the editor shows the time as 8:20.

# Bài tập

- Viết chương trình C thực hiện công việc sau:
  - Nhập vào 3 điểm Toán, Lý, Hóa
  - Tính và in ra điểm trung bình 3 môn (làm tròn đến 2 chữ số thập phân)
  - Tìm và in ra điểm lớn nhất (sử dụng biểu thức điều kiện)
  - Tìm và in ra điểm nhỏ nhất (sử dụng biểu thức điều kiện)