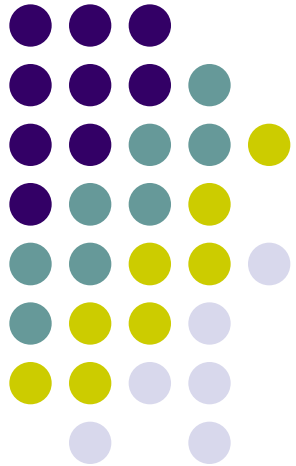
LẬP TRÌNH CĂN BẢN

Phần 2 - Chương 2

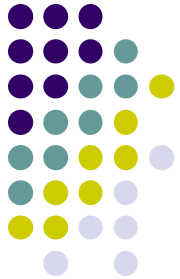
CÁC THÀNH PHẦN CƠ BẢN CỦA NGÔN NGỮ C



Nội dung chương này

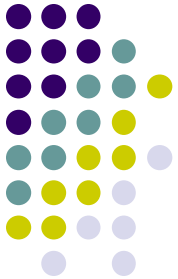
- Bộ chữ viết trong C
- Các từ khóa
- Cặp dấu ghi chú thích
- Các kiểu dữ liệu sơ cấp chuẩn
- Tên và hằng
- Biến và biểu thức
- Cấu trúc của một chương trình C

Bộ chữ viết trong C



- Bộ chữ viết trong ngôn ngữ C bao gồm các ký tự sau:
  - 26 chữ cái latinh lớn A,B,C...Z
  - 26 chữ cái latinh nhỏ a,b,c ...z.
  - 10 chữ số thập phân 0,1,2...9.
  - Các ký hiệu toán học: +, -, *, /, =, <, >, (,)
  - Các ký hiệu đặc biệt: :. , ; " ' _ @ # $! ^ [ ] { } ...
  - Dấu cách hay khoảng trống.
- ***Phân biệt chữ in hoa và in thường***

Các từ khóa trong C



- Từ khóa là các từ dành riêng của C.
- **Ta không được dùng từ khóa để đặt cho các tên của riêng mình.**

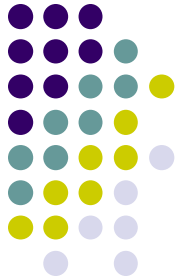
asm	auto	break	case	cdecl	char
class	const	continue	_cs	default	delete
do	double	_ds	else	enum	_es
extern	_export	far	_fastcall	float	for
friend	goto	huge	if	inline	int
interrupt	_loadds	long	near	new	operator
pascal	private	protected	public	register	return
_saveregs	_seg	short	signed	sizeof	_ss
static	struct	switch	template	this	typedef
union	unsigned	virtual	void	volatile	while

Cặp dấu chú thích (comment)

```
#include <stdio.h>
#include<conio.h>
int  main (){
      char ten[50];   /* khai bao bien ten
                         kieu char 50 ky tu */
      printf("Xin cho biet ten cua ban !");
      scanf("%s",ten);  /*Doc vao 1 chuoi la ten ban*/
      printf("Xin chao ban %s\n ",ten);
      //Dung chuong trinh, cho go phim
      getch();
      return 0;
}
```

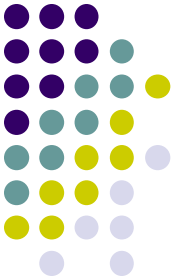
- Khi biên dịch các phần chú thích bị bỏ qua
- Dùng /* và */: chú thích dài nhiều dòng
- Dùng //: chú thích chỉ 1 dòng

Các kiểu dữ liệu sơ cấp chuẩn trong C



- Kiểu số nguyên (integer)
- Kiểu số thực (real)

Kiểu số nguyên



- Được dùng để lưu các giá trị nguyên hay còn gọi là kiểu đếm được.
  - ***Kiểu số nguyên 1 byte (8 bits)***

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned char	Từ **0** đến **255** (*tương đương 256 ký tự trong bảng mã ASCII*)
2	char	Từ **-128** đến **127**

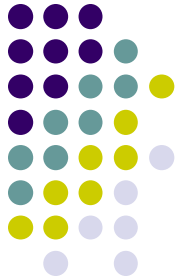
  - ***Kiểu số nguyên 2 bytes (16 bits)***

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	enum	Từ -32,768 đến 32,767
2	unsigned int	Từ **0** đến **65,535**
3	short int	Từ -32,768 đến 32,767
4	int	Từ -32,768 đến 32,767

  - ***Kiểu số nguyên 4 byte (32 bits)***

STT	Kiểu dữ liệu	Miền giá trị (Domain)
1	unsigned long	Từ **0** đến **4,294,967,295**
2	long	Từ **-2,147,483,648** đến **2,147,483,647**

Kiểu số thực

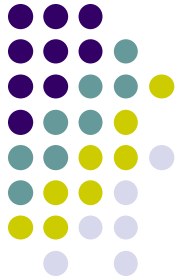


- Được dùng để lưu các số thực hay các số có dấu chấm thập phân

STT	Kiểu dữ liệu	Kích thước (Size)	Miền giá trị (Domain)
1	float	4 bytes	Từ $3.4 * 10^{-38}$ đến $3.4 * 10^{38}$
2	double	8 bytes	Từ $1.7 * 10^{-308}$ đến $1.7 * 10^{308}$
3	long double	10 bytes	Từ $3.4 * 10^{-4932}$ đến $1.1 * 10^{4932}$

- Kiểu **void**
  - Mang ý nghĩa là kiểu rỗng không chứa giá trị gì cả
  - **Ví dụ:** void main(){
    ….}

Dùng sizeof()

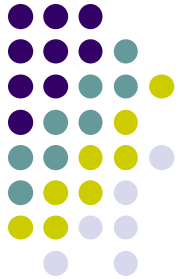


- Kích thước 1 kiểu có thể được xác định lúc chạy chương trình (runtime), dùng sizeof:
  - **Ví dụ:**

    sizeof(double) =>8(byte)

    sizeof(long double)=>10(byte)

Tên và hằng trong C

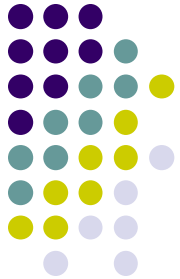


- *Tên* (identifier)
  - Được dùng để đặt cho chương trình, hằng, kiểu, biến, chương trình con, ...
  - Có 2 loại:
    - **Tên chuẩn**: là tên do C đặt sẵn như tên kiểu: int, char, float,…; tên hàm: sin, cos...
    - **Tên do người lập trình tự đặt.**

Chú ý khi đặt tên

```
main        ≠   Main
myVariable  ≠   MyVariable
```

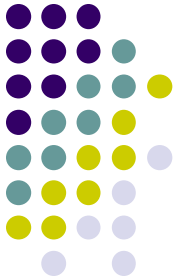
Tên do người lập trình tự đặt



- **Ví dụ:**
  - Tên đặt hợp lệ: Chieu_dai, Chieu_Rong, Chu_Vi
  - Tên không hợp lệ: Do Dai, 12A2

- Phải tuân thủ quy tắc:
  - Sử dụng bộ chữ cái, chữ số và dấu gạch dưới (_)
  - Bắt đầu bằng một chữ cái hoặc dấu gạch dưới.
  - Không có khoảng trống ở giữa tên.
  - Không được trùng với từ khóa.
  - Độ dài tối đa của tên là 32 ký tự, tuy nhiên cần đặt sao cho rõ ràng, dễ nhận biết và dễ nhớ.
  - Không cấm việc đặt tên trùng với tên chuẩn nhưng khi đó ý nghĩa của tên chuẩn không còn giá trị nữa.

Hằng (Constant)



- Là đại lượng không đổi trong suốt quá trình thực thi chương trình

```
...
const double pi = 3.14159254;
const int maxval = 127;
```

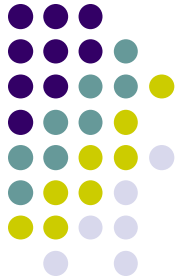
=> không thể gán lạl giá trị cho hằng

```
...
pi = 5;
```

gcc → „warning assignment of read-only variable **pi**“

- Hằng có thể là:
  - 1 con số
  - 1 ký tự
  - 1 chuỗi ký tự

Hằng số thực



- Giá trị kiểu: float, double, long double
- 2 cách thể hiện
  - ***Cách 1*:** viết thông thường
    - **Ví dụ:** 123.34 -223.333 3.00 -56.0
  - ***Cách 2*:** viết theo số mũ hay số khoa học
    - Một số thực được tách làm 2 phần *(phân cách bởi e/E)*
      - **Phần giá trị:** như cách 1
      - **Phần mũ:** là một số nguyên
    - **Ví dụ:**

1234.56e-3 = 1.23456 (là số $1234.56*10^{-3}$)

-123.45E4 = -1234500 (là $-123.45*10^{4}$)

Hằng số nguyên (1)

- Hằng số nguyên 2 byte (int) hệ thập phân
  - Sử dụng 10 ký số 0..9
  - **Ví dụ:**

    123 (một trăm hai mươi ba)

    -242 (trừ hai trăm bốn mươi hai)
- Hằng số nguyên 2 byte (int) hệ bát phân
  - Sử dụng 8 ký số 0..7
  - Cách biểu diễn: 0<các ký số từ 0 đến 7>
  - Số bát phân : $0d_nd_{n-1}d_{n-2}\ldots d_1d_0$ (d_i có giá trị từ 0..7)

    => giá trị: $\sum_{i=0}^{n} d_i * 8^i$

  - **Ví dụ:**

    $020=2*8^1 + 0*8^0 = (16)_{10}$

Hằng số nguyên (2)

- *Hằng số nguyên 2 byte (int) hệ thập lục phân*
  - **Là kiểu số nguyên dùng:**
    - 10 ký số 0..9 và
    - 6 ký tự A, B, C, D, E ,F
  - **Cách biểu diễn**:

    0x<các ký số từ 0 đến 9 và 6 ký tự từ A đến F>
  - **Số thập lục phân : $0xd_nd_{n-1}d_{n-2}\ldots d_1d_0$**

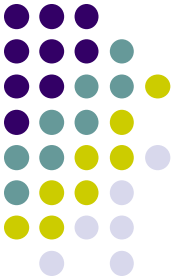
    => Giá trị thập phân= $\sum_{i=0}^{n} d_i * 16^i$
  - **Ví dụ:**

    $0x345=3*16^2 + 4*16^1 + 5*16^0 = (837)_{10}$

    $0x2A9= 2*16^2 + 10*16^1 + 9*16^0= (681)_{10}$

Ký tự	giá trị
A	10
B	11
C	12
D	13
E	14
F	15

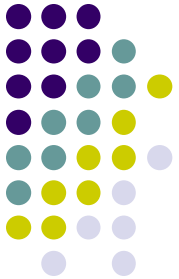
Hằng số nguyên (3)



- Ví dụ: Kết quả của chương trình sau là gi?

```
#include <conio.h>
#include <stdio.h>
void main(){
        int a=010;  //octal
        int b=10;   //decimal
        int c=0x10; //hexadecimal

        clrscr();
        if(a==b)printf("010==10 is wrong!");
        if(a==8)printf("\n010==8 is correct!");
        if(c==16)printf("\n0x10==16 is correct!");
        getch();
}
```

Hằng số nguyên (4)

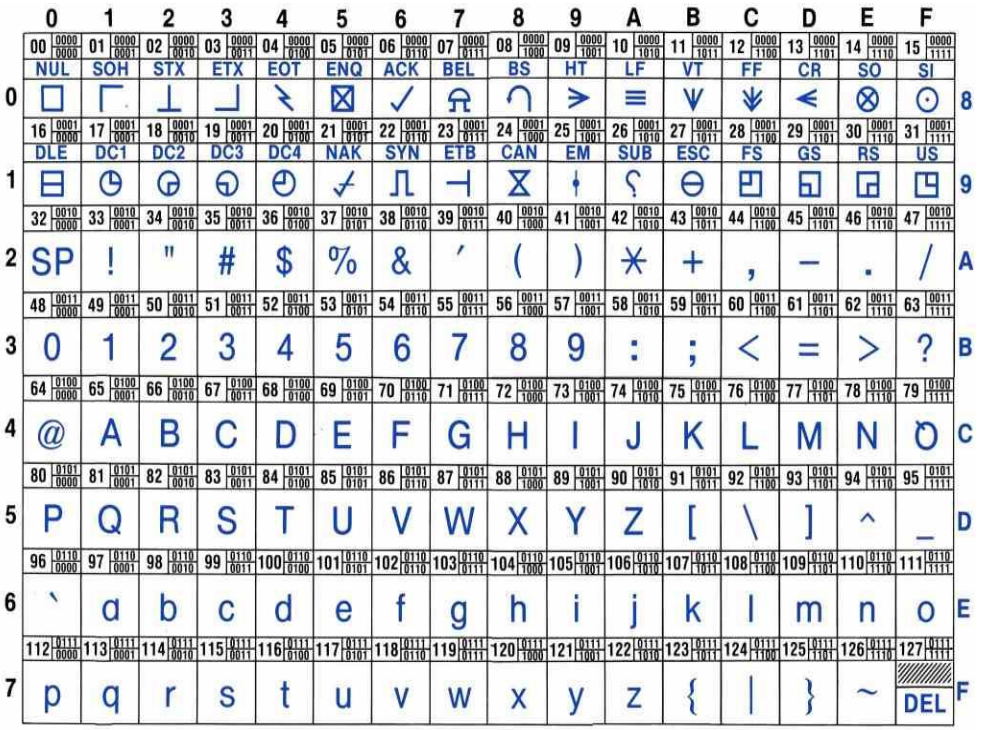


- ***Hằng số nguyên 4 byte (long)***
  - Được biểu diễn như số int trong hệ thập phân nhưng kèm theo ký tự **l** hoặc L.
  - **Ví dụ:**

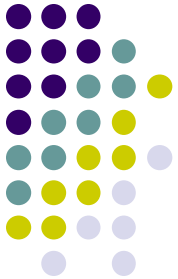
45345L *hay* 45345l *hay* 45345

Hằng ký tự (char)

- *Ví dụ*: ‘a’, ‘A’, ‘0’, ‘9’
- Là 1 ký tự được viết trong cặp dấu nháy đơn (‘).
- Mỗi một ký tự tương ứng với 1 giá trị trong bảng mã ASCII.
- Hằng ký tự cũng được xem như trị số nguyên.
- Chúng ta có thể thực hiện các phép toán số học trên 2 ký tự (dùng giá trị ASCII của chúng)
- **ASCII** = American Standard Code for Information Interchange



Hằng chuỗi ký tự



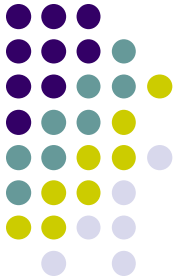
- ***Ví dụ*: "Ngon ngu lap trinh C"**
- Là 1 chuỗi hay 1 xâu ký tự được đặt trong cặp dấu nháy kép (").

- ***Chú ý*:**
  - "" : chuỗi rỗng - không có nội dung
  - Khi lưu trữ trong bộ nhớ, một chuỗi được kết thúc bằng ký tự NULL ('\0': mã Ascii là 0).
  - Để biểu diễn ký tự đặc biệt bên trong chuỗi ta phải thêm dấu \ phía trước.
    - ***Ví dụ*:**

      Viết **"I\'m a student"** cho **"I'm a student"**

      Viết **"Day la ky tu \"dac biet\""** cho **"Day la ky tu "dac biet""**

Biến và Biểu thức (variable and expression)



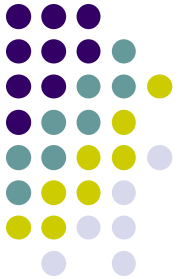
```
int    a, b, c;
long int    chu_vi;
float     nua_chu_vi;
double  dien_tich;
```

- Biến dùng để chứa dữ liệu trong quá trình thực hiện chương trình.
- Giá trị của biến có thể bị thay đổi.
- Cú pháp khai báo biến:

  <Kiểu dữ liệu> Danh sách các tên biến cách nhau bởi dấu phẩy;

```
Type        name of variable
int         a;
double      f;
boolean     error;
```

Khởi tạo giá trị cho biến lúc khai báo



- **Ví dụ:**

```
int anInt = 4;
double pi = 3.141592654;
```

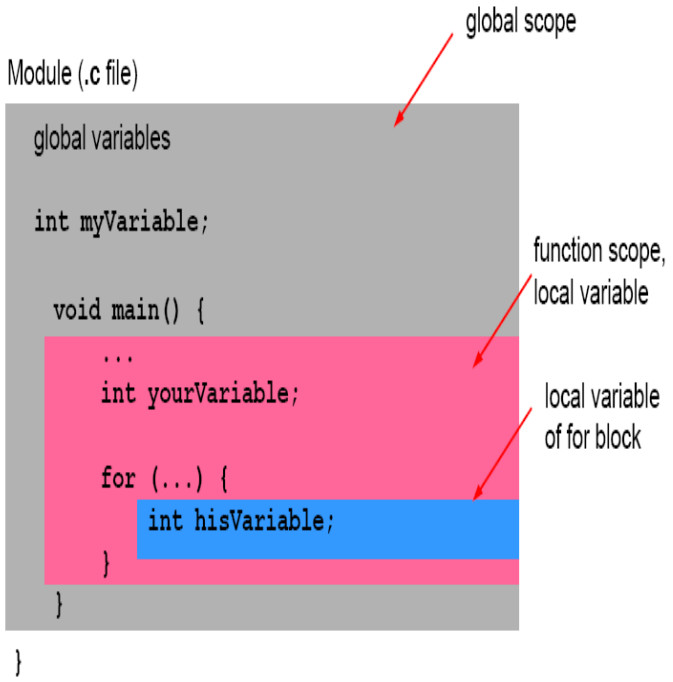
- Cách viết giá trị cho biết luôn kiểu của nó:

Literal	Type	
178	int	
8864L	long	
37.266	double	
37.266d	double	
87.383f	float	
26.77e3	double	26.77×10^3
1.25e-1f	float	1.25×10^{-1}
'c'	char	

- **Chú ý:** 8864L có kiểu long, còn 8864 có kiểu int

Vị trí khai báo biến (1)

- Biến ngoài
  - Được đặt bên ngoài tất cả các hàm
  - Ảnh hưởng đến toàn bộ chương trình (biến toàn cục)

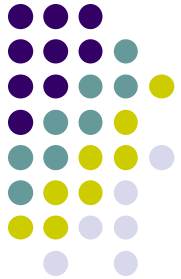


Vị trí khai báo biến (2)

- **Biến trong**
  - Được đặt bên trong hàm, chương trình chính hay một khối lệnh
  - Ảnh hưởng đến hàm, chương trình hay khối lệnh chứa nó (biến cục bộ).

```
#include <stdio.h>
#include <conio.h>
int bienngoai;    /*khai bao bien ngoai*/
int main (){
  int j,i;       /*khai bao bien ben trong*/
  clrscr();
  i=1; j=2;
  bienngoai=3;
  printf("\n Gia7 tri cua i la %d",i);
  printf("\n Gia tri cua j la %d",j);
  printf("\n Gia tri cua bienngoai la %d",bienngoai);
  getch();
  return 0;
}
```

Biểu thức (1)



- Ví dụ:

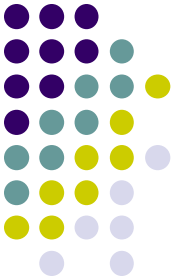
  (-b + sqrt(Delta))/(2*a)

- Biểu thức là một sự kết hợp giữa
  - Các toán tử (operator) và
  - Các toán hạng (operand)
- Các loại toán tử trong C
  - Toán tử số học
  - Toán tử quan hệ và logic
  - Toán tử Bitwise
  - Toán tử ?
  - Toán tử con trỏ & và *
  - Toán tử dấu phẩy

Các toán tử số học (1)

Toán tử	Ý nghĩa
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy phần dư
--	Giảm 1 đơn vị
++	Tăng 1 đơn vị

Các toán tử số học (2)



+	Addition	x = 3+5 ⇒ 8
-	Subtraction	x = 5-3 ⇒ 2
*	Multiplication	x = 5*3 ⇒ 15
/	Division	x = 14/3 ⇒ 4 (integer division if both operands int)
%	Modulo	x = 14%3 ⇒ 2

- **Tăng và giảm (++ & --)**

  ++x hay x++ giống x = x + 1

  --x hay x-- giống x = x – 1

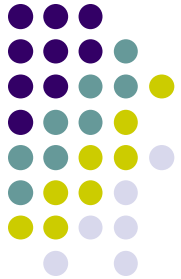
- **Tuy nhiên:**

```
x = 10;
y = ++x;  //y = 11, x=11
```

- **Còn:**

```
x = 10;
y = x++;  //y = 10, x=11
```

Các toán tử số học (3)



```
int x = 3;
int y;

y = x++;
printf("%d\n", x); ⇒ 4
printf("%d\n", y); ⇒ 3
```

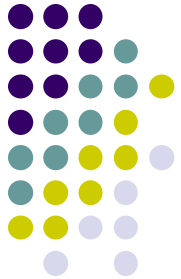
```
int x = 3;
int y;

y = ++x;
printf("%d\n", x); ⇒ 4
printf("%d\n", y); ⇒ 4
```

- Đâu là sự khác nhau?

  x++ trả về giá trị hiện hành của x và sau đó tăng x

  ++x tăng x trước và sau đó trả về giá trị mới của x

Biểu thức Boolean (boolean expression)



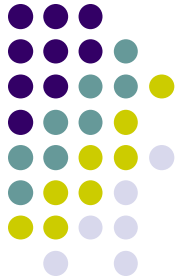
- **Chú ý!** Không có kiểu Boolean rõ ràng trong C (điều này đã được giới thiệu ở C99). Thay vào đó C dùng các giá trị nguyên để tượng trưng cho giá trị Boolean, với qui ước:

false	Giá trị 0
true	Bất kỳ giá trị nào ngoại trừ 0

- **Chú ý!** C dùng "=" cho phép gán, và dùng "==" cho phép so sánh. Nó trả về 1 nếu bằng và 0 nếu ngược lại

```
printf("%d\n", 1==2);   ⇒ 0
printf("%d\n", 1==1);   ⇒ 1
```

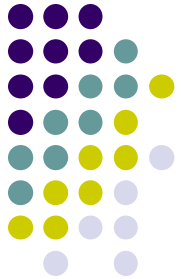
Các toán tử quan hệ và các toán tử Logic (1)



- **Các phép so sánh sau tạo ra các biểu thức logic có giá trị kiểu Boolean**

Toán tử	Ý nghĩa
Các toán tử quan hệ	
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Khác
Các toán tử Logic	
&&	AND
||	OR
!	NOT

Các toán tử quan hệ và các toán tử Logic (2)

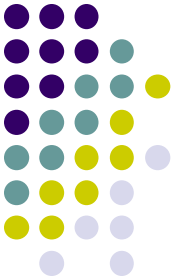


- **Ví dụ:**

Operator	Meaning	Example
==	equal	x == 3
!=	not equal	x != y
>	greater	4 > 3
<	less	x < 3
>=	greater or equal	x >= y
<=	less or equal	x <= y

- Các biểu thức logic trả về
  - **0** nếu **false(sai)**
  - **1** nếu **true(đúng)**

Các toán tử quan hệ và các toán tử Logic (3)



- Bảng chân trị cho các toán tử Logic

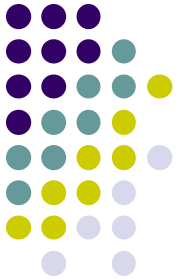
p	q	p&&q	p||q	!p
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

- Thứ tự ưu tiên

Cao nhất:	!			
	>	>=	<	<=
	==	!=		
	&&			
Thấp nhất:	||			

- **Ví dụ:** 10>5&&!(10<9)||3<=4 => đúng (1)

Các toán tử Bitwise(cho lop nghi hoc roi)



- Toán tử Bitwise giúp kiểm tra, gán hay thay đổi các bit thật sự trong 1 byte của word.
- Chỉ dùng cho kiểu char và int.

Toán tử	Ý nghĩa
&	AND
|	OR
^	XOR
~	NOT
>>	Dịch phải
<<	Dịch trái

Bảng chân trị của toán tử ^ (XOR)

p	q	p^q
0	0	0
0	1	1
1	0	1
1	1	0

Toán tử ?

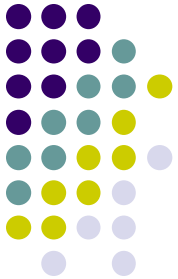
- Toán tử ? thực hiện như lệnh if-else.
- Cú pháp:
  - E1 ? E2 : E3
- **Ví dụ:** X = (10 > 9) ? 100 : 200;

  =>X=100

  X = (10 >15)? 100 : 200;

  =>X=200

Toán tử con trỏ & và *



- Ví dụ:

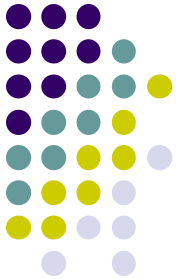
```
int *p;              //con tro so nguyen
int count=5, x;
p = &count;
```

=>Đặt vào biến m địa chỉ bộ nhớ của biến count

- Toán tử * trả về nội dung của ô nhớ mà một con trỏ đang chỉ vào
  - Ví dụ:

```
x = *p; // x=5
```

Toán tử dấu phẩy



- **Ví dụ**:
  - x = (y=3,y+1);
  - Trước hết gán 3 cho y rồi gán 4 cho x.

- Được sử dụng để kết hợp các biểu thức lại với nhau.
- Bên trái của dấu (,) luôn được xem là kiểu void.
- Biểu thức bên phải trở thành giá trị của tổng các biểu thức được phân cách bởi dấu phẩy.

Tổng kết về độ ưu tiên

- **Tổng kết về độ ưu tiên**

Cao nhất	() []
	! ~ ++ -- (Kiểu) * &
	* / %
	+ -
	<< >>
	< <= > >=
	&
	^
	|
	&&
	||
	?:
	= += -= *= /=
Thấp nhất	,

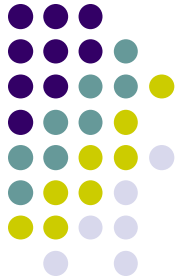
Phép gán được viết gọn lại

x= x <phép toán> y;

có thể được viết gọn lại (short form):

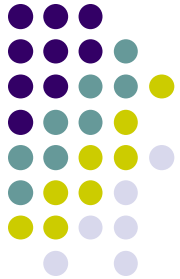
	short form	expanded form
+=	`x += y;`	`x = x + y;`
-=	`x -= y;`	`x = x - y;`
*=	`x *= y;`	`x = x * y;`
/=	`x /= y;`	`x = x / y;`
%=	`x %= y;`	`x = x % y;`

Các tập tin thư viện thông dụng



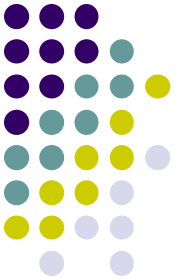
- **stdio.h**: Định nghĩa các hàm vào/ra chuẩn (standard input/output):printf(), scanf(), getc(), putc(), gets(), puts(), fflush(), fopen(), fclose(), fread(), fwrite(), getchar(), putchar(), getw(), putw()…
- **conio.h**: Định nghĩa các hàm vào ra trong chế độ DOS: clrscr(), getch(), getche(), getpass(), cgets(), cputs(), putch(), clreol(),…
- **math.h**: Định nghĩa các hàm tính toán: abs(), sqrt(), log(). log10(), sin(), cos(), tan(), acos(), asin(), atan(), pow(), exp(),…
- **alloc.h**: Định nghĩa các hàm liên quan đến việc quản lý bộ nhớ: calloc(), realloc(), malloc(), free(), farmalloc(), farcalloc(), farfree(), …
- **io.h**: Định nghĩa các hàm vào ra cấp thấp: open(), _open(), read(), _read(), close(), _close(), creat(), _creat(), creatnew(), eof(), filelength(), lock(),…
- **graphics.h**: Định nghĩa các hàm liên quan đến đồ họa: initgraph(), line(), circle(), putpixel(), getpixel(), setcolor(), …

Cấu trúc của 1 chương trình C (1)



- Cấu trúc một chương trình C
- Tiền xử lý và biên dịch
- Prototype
- Các tập tin thư viện thông dụng

Cấu trúc của 1 chương trình C (2)



Các chỉ thị tiền xử lý

Định nghĩa kiểu mới

Prototype

Khai báo biến ngoài

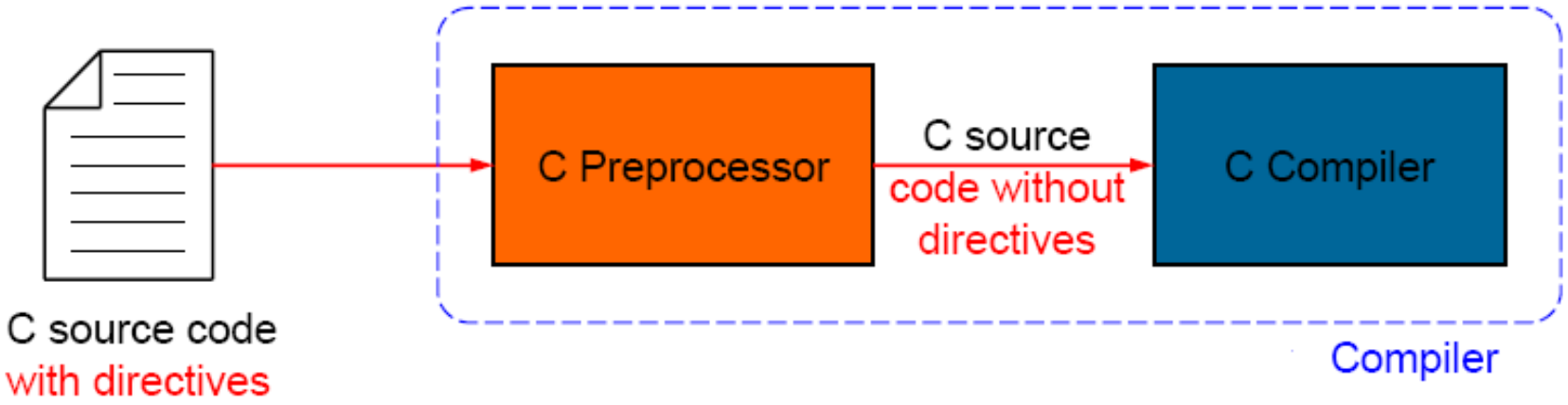
```
#include <stdio.h>
#include <conio.h>

#define Pi 3.14
#define BEGIN {
#define END }
#define max(a,b)(a>b)?a:b

typedef float sothuc;

float cv_dtron(float r);
float dt_dtron(float r);

char *tb="Tinh CV * DT";
```

```
void main(){
  float bk;
  sothuc a=5, b=7;
  clrscr();
  printf("Chuong trinh %s\n",tb);
  printf("So max =%.2f\n",max(a,b));
  printf("Nhap ban kinh r=");scanf("%f",&bk);
  printf("Chu vi dtron=%.2f\n",cv_dtron(bk));
  printf("Dien tich =%.2f\n",dt_dtron(bk));
  getch();
}
float  cv_dtron(float r){
     return 2*Pi*r;
}
float  dt_dtron(float r)BEGIN
     return Pi*r*r;
END
```

Chương trình chính

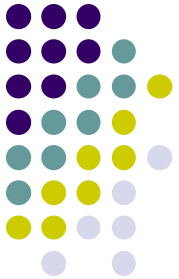
Cài đặt các hàm

Tiền xử lý và biên dịch (preprocess and compile)



- Các chỉ thị định hướng (directive):
  - #include…, #define…
  - Có thể chứa các lệnh phức tạp như if-else.
- Bộ tiền xử lý (preprocessor) sẽ ***thông dịch*** các directive và xóa bỏ nó trước khi cung cấp cho trình biên dịch C.

Chia chương trình ra các module (1)



- 1 chương trình phức tạp có thể được chia ra vài module

```c
#include <stdio.h>
/* Prototypes */
int foo(int a, int b);
int bar(int a, int *result);

main() {
   int myVar=2,yourVar=3,res;
   res = foo(myVar, yourVar);
   printf("%d\n", res);
   bar(myVar, &res);
   printf("%d\n", res);
}

int foo(int a, int b) {
   return a+b;
}

void bar(int a, int *result) {
   *result = a;
}
```

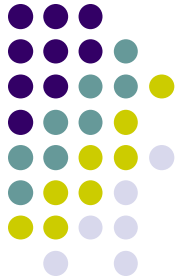
```c
#include <stdio.h>

main() {
    int myVar=2,yourVar=3,res;
    res = foo(myVar, yourVar);
    printf("%d\n", res);
    bar(myVar, &res);
    printf("%d\n", res);
}
```

testmodule.c

```c
int foo(int a, int b) {
   return a+b;
}

void bar(int a, int *result) {
   *result = a;
}
```

mymodule.c

Chia chương trình ra các module (2)



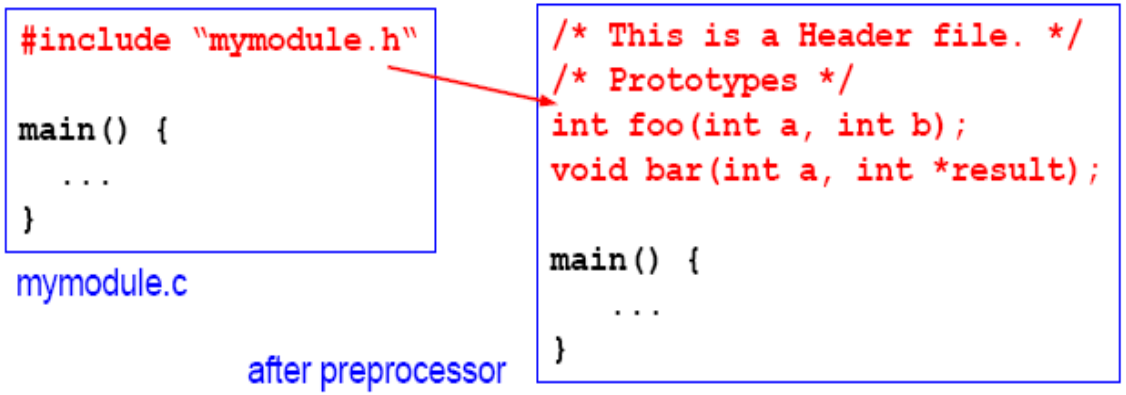
- Vấn đề: **testmodule.c** phải biết các prototype của **foor** và **bar**.
- **Giải pháp 1 (tệ):**
  - Chèn tay các prototype vào các file .c có dùng nó.
  - **Bất lợi:** Mỗi khi prototype bị thay đổi => phải chỉnh lại prototype trong tất cả các file .c dùng nó.
- **Giải pháp 2 (tốt):**
  - Lưu các prototype vào 1 file riêng biệt mymodule.h (h: header).
  - Dùng **#include mymodule.h** ở đầu các chương trình có dùng nó.

```
/* This is a header file. */
/* Prototypes */
int foo(int a, int b);
void bar(int a, int *result);
```

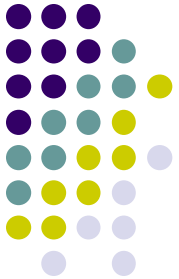
mymodule.h

#include

- Với #include, bộ tiền xử lý sẽ thêm và thay thế token **#include filename** bằng nội dung của **filename**.



- Các header file sẽ được tìm ở đâu?
  - #include <file.h>: tìm file.h trong thư mục đã được xác định trong INCLUDE DIRECTORIES. Hoặc trong /usr/include (linux)
  - #include "C:\\TC\\file.h": tìm file.h trong đường dẫn



Hết chương