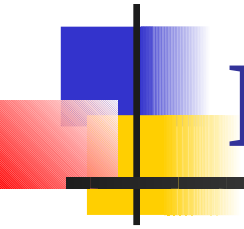


NGÔN NGỮ LẬP TRÌNH C

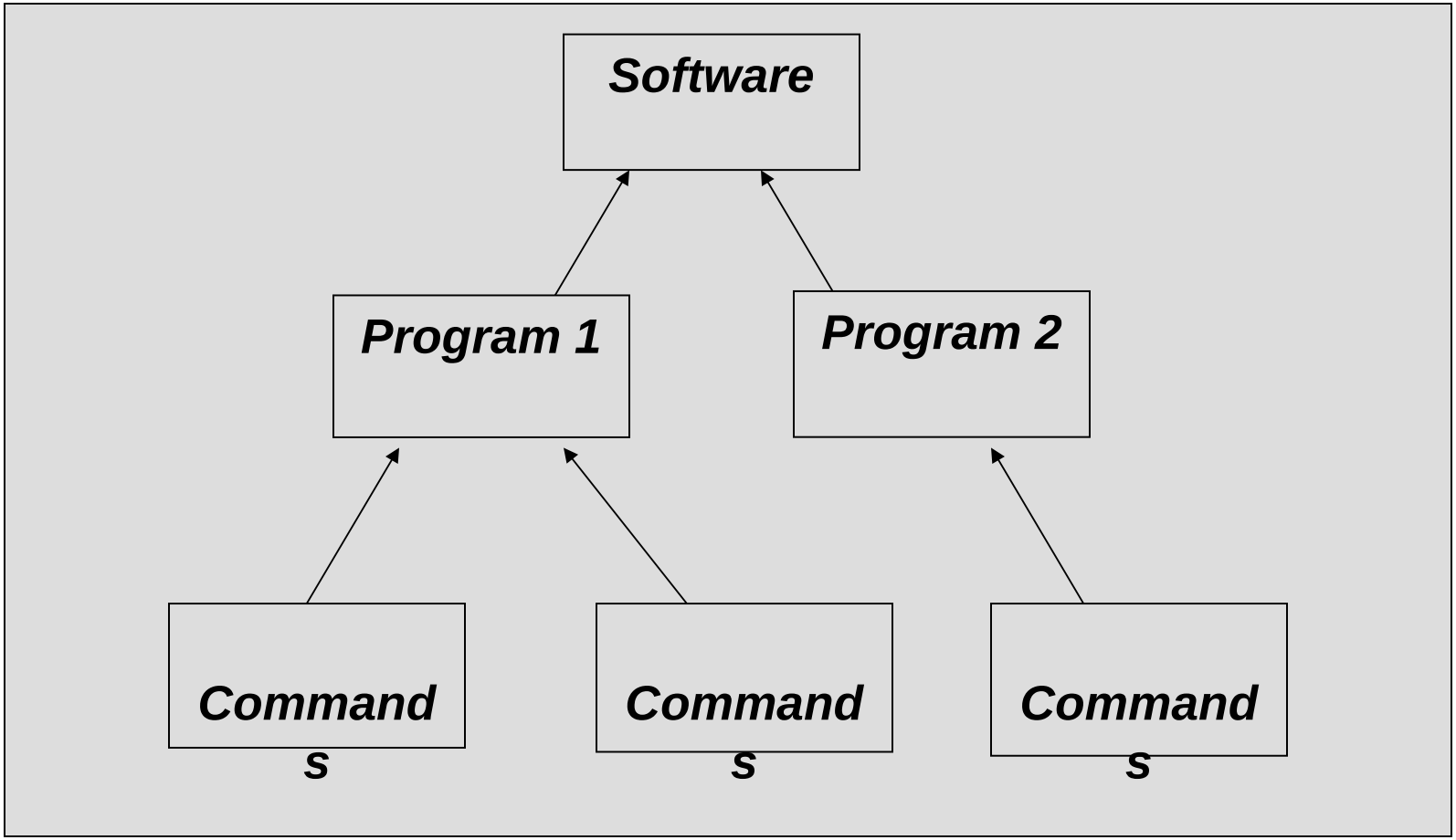
Chương 1

TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

Mục tiêu của bài giảng

- Phân biệt sự khác nhau giữa Câu lệnh, Chương trình và Phần mềm
- Biết được quá trình hình thành ngôn ngữ C
- Lựa chọn được một số trình biên dịch và công cụ hỗ trợ lập trình C.
- Nắm được các thành phần cơ bản của C.
- Biết cách viết, biên dịch và chạy một chương tình C đơn giản.

Phần mềm, chương trình, câu lệnh



Lịch sử ngôn ngữ C

- Lịch sử ngôn ngữ C
  - Ra đời vào đầu những năm 70 của thế kỉ XX, do Dennish Ritchie phát triển dựa trên ngôn ngữ BCPL của Martin Richards.
  - Mục đích ban đầu của C là để viết hệ điều hành Unix.
  - Được đặt tên C vì trước đó đã có ngôn ngữ B tại Bell.
  - C có nhiều ưu điểm đặc biệt là tính mềm dẻo cao nên nhanh chóng trở thành ngôn ngữ chính thống.
  - Có nhiều phiên bản và tình dịch C khác nhau:
    - ANSI C.
    - ISO C
    - Turbo C

Một số ưu điểm của C

- Là ngôn ngữ lập trình đa năng, mạnh và mềm dẻo.
- Chương trình viết bằng C chạy nhanh hơn so với chương tình viết bằng Pascal.
- Thường được sử dụng để lập trình hệ thống (hệ điều hành ..)
- Là ngôn ngữ dễ thích nghi với nhiều môi trường khác nhau.
- Là ngôn ngữ có cấu trúc module (chương trình = các hàm).

Ngôn ngữ cấp trung

Ngôn ngữ cấp cao

C

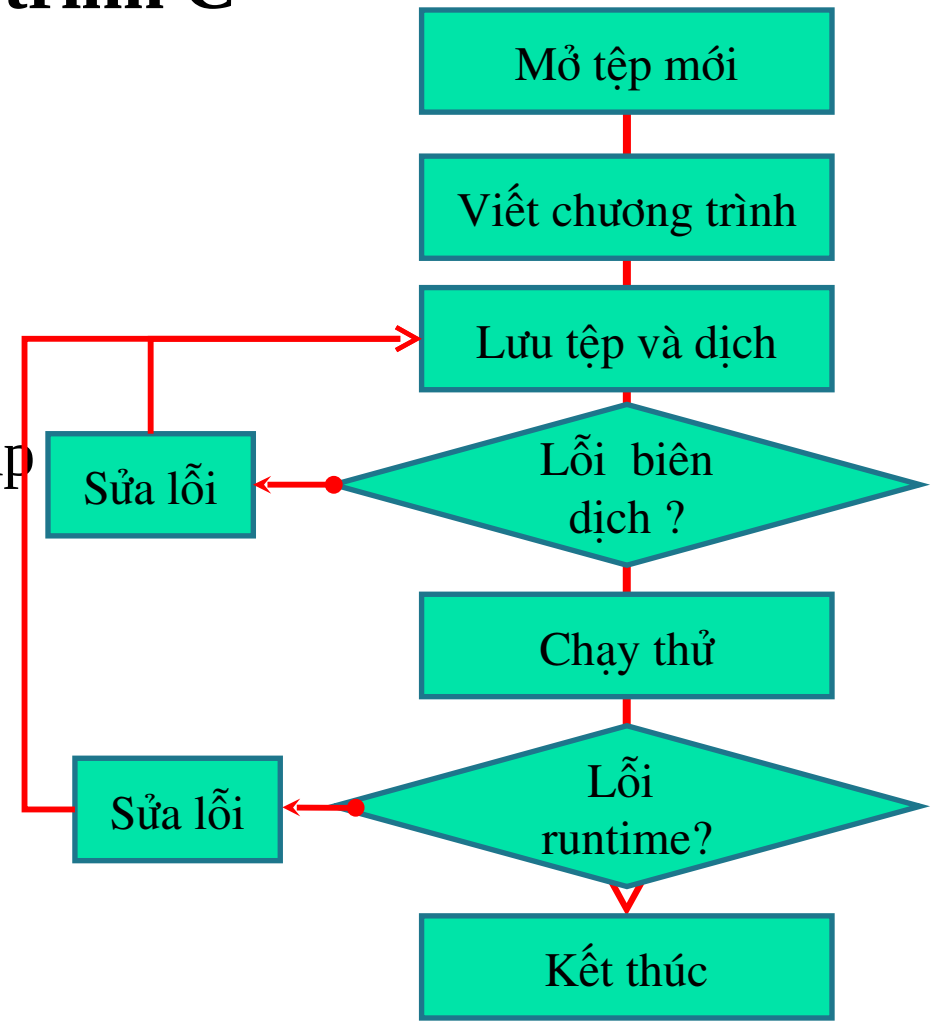
Ngôn ngữ hợp ngữ

Các bộ trình biên dịch C

- Turbo C → Borland C → Borland C++ → Borland C builder
- Microsoft C → Visual C++
- C Free
- Ngoài ra còn có các IDE (intergrated Development Eniroment): Visual Studio, Eclipse, …

Các bước cần thực hiện khi viết 1 Chương trình C

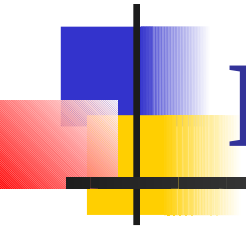
1. Mở tệp mới (File\New)
2. Viết chương trình
3. Lưu tệp và dịch
4. Nếu có lỗi thì
   - Sửa lỗi chính tả, cú pháp
   - Quay lại bước 3
1. Chạy thử
2. Nếu có lỗi thì
   - Dò và sửa lỗi
   - Quay lại bước 3
1. Kết thúc



Chương trình Hello

- Lập chương trình có chức năng viết lên màn hình câu chào “Hello, world!”

```
#include <stdio.h>
void main(){
        printf(“\nHello, world!”);
}
```



NGÔN NGỮ LẬP TRÌNH C

Các thành phần cơ bản của ngôn ngữ lập trình C

Bộ ký tự

- Bộ chữ viết trong ngôn ngữ C bao gồm những ký tự, ký hiệu sau: (*phân biệt chữ in hoa và in thường*):
- 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.

Các từ khóa

- C có 32 từ khóa chuẩn và các từ khóa mở rộng bao gồm:

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 ghi chú thích /*…*/

- Trong chương trình C, nội dung chú thích phải được viết trong cặp dấu /* …. */
- Ví dụ:

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    /* Xuat chuoi ra man hinh */
    printf("Hello C-Free!\n");

    /* In tong cua hai so ra man hinh */
    printf("Total: %d\n",total(10, 20));
    return 0;
}
/* Ham tinh tong cua hai so */
int total(int a, int b){
    return (a+b);
}
```

Dấu chấm phẩy và cặp { }

- Câu lệnh và dấu chấm phẩy:
- Nói chung, mỗi câu lệnh đơn nên viết trên một dòng.
- Kết thúc câu lệnh bằng dấu chấm phẩy ;
- Một số chỉ dẫn (không phải câu lệnh) không cần dấu ;
  - #include “stdio.h”
  - #include “conio.h”
- Cặp { } có giá trị bắt đầu và kết thúc một khối lệnh

Kiểu dữ liệu

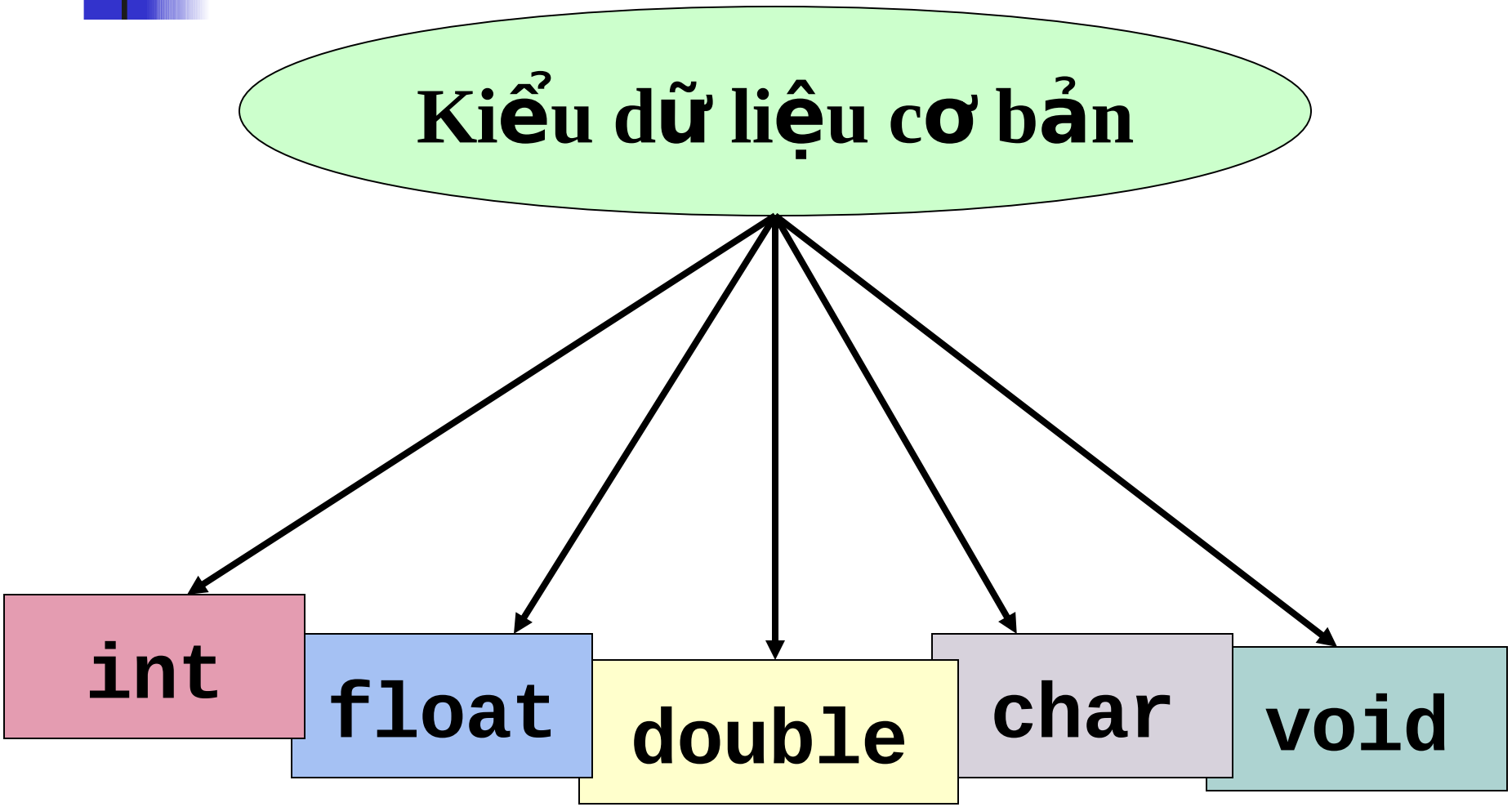
- Các kiểu dữ liệu khác nhau được lưu trữ trong biến là:
  - Số (Numbers)
    - Số nguyên.
      Ví dụ : 10 hay 178993455
    - Số thực.
      Ví dụ, 15.22 hay 15463452.25
    - Số dương
    - Số âm
  - Tên. Ví dụ : John
  - Giá trị logic :
    Ví dụ : Y hay N hoặc T hay F

Kiểu dữ liệu (tt.)

- Kiểu dữ liệu mô tả loại dữ liệu sẽ được lưu trong biến
- Tên biến đặt sau kiểu dữ liệu
- Ví dụ : tên biến "varName" đứng sau kiểu dữ liệu "int"

kiểu dữ liệu	tên biến
int	varName

Kiểu dữ liệu cơ bản



Kiểu số nguyên (int)

- Lưu trữ dữ liệu số

  int num;
- Không thể lưu trữ bất cứ kiểu dữ liệu nào khác như “Alan” hoặc “abc”
- Chiếm 16 bits (2 bytes) bộ nhớ
- Biểu diễn các số nguyên trong phạm vi -32768 tới 32767
- Ví dụ : 12322, 0, -232

Kiểu số thực (float)

- Lưu trữ dữ liệu số chứa phần thập phân

  float num;
- Có độ chính xác tới 6 con số
- Chiếm 32 bits (4 bytes) bộ nhớ
- 3.4E-38 đến 3.4E+38 (10 mũ dương 38)
- Ví dụ : 23.05, 56.5, 32

Kiểu số thực (double)

- Lưu trữ dữ liệu số chứa phần thập phân

  double num;
- Có độ chính xác tới 10 con số
- Chiếm 64 bits (8 bytes) bộ nhớ
- 1.7E-308 đến 1.7E+308
- Ví dụ : 23.05, 56.5, 32

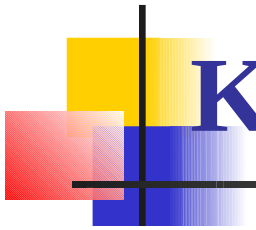
Kiểu ký tự (char)

- Lưu trữ một ký tự đơn

  char gender;

  gender='M';

- Chiếm 8 bits (1 byte) bộ nhớ
- Ví dụ: ‘a’, ‘m’, ‘$’ ‘%’ , ‘1’, ’5’



Kiểu void

- Không lưu bất cứ dữ liệu gì
- Báo cho trình biên dịch không có giá trị trả về

Những kiểu dữ liệu dẫn xuất

Bộ bổ từ (Modifiers) kiểu dữ liệu	+	Kiểu dữ liệu cơ bản	=	Kiểu dữ liệu dẫn xuất
unsigned	+	int	=	unsigned int (chỉ là số dương)
short	+	int	=	short int (chiếm ít bộ nhớ hơn int)
long	+	int/double	=	Long int /longdouble (chiếm nhiều bộ nhớ hơn int/double)

Các kiểu dữ liệu signed và unsigned

- Kiểu unsigned chỉ rõ rằng một biến chỉ có thể nhận giá trị dương

```
unsigned int varNum;
varNum=23123;
```

- varNum được cấp phát 2 bytes
- Bổ từ unsigned có thể được dùng với kiểu dữ liệu int và float
- Kiểu unsigned int hỗ trợ dữ liệu trong phạm vi từ 0 đến 65535

Những kiểu dữ liệu long (dài) và short (ngắn)

- **short int** chiếm giữ 8 bits (1 byte)
  - Cho phép số có phạm vi từ -128 tới 127
- **long int** chiếm giữ 32 bits (4 bytes)
  - -2,147,483,648 và 2,147,483,647
- **long double** chiếm 128 bits (16 bytes)

Kiểu dữ liệu & phạm vi giá trị

Kiểu	Dung lượng tính bằng bit	Phạm vi
char	8	-128 tới 127
Unsigned char	8	0 tới 255
signed char	8	-128 tới 127
int	16	-32,768 tới 32,767
unsigned int	16	0 tới 65,535
signed int	16	Giống như kiểu int
short int	16	Giống như kiểu int
unsigned short int	16	0 tới 65, 535

Kiểu dữ liệu & phạm vi giá trị (tt.)

Kiểu	Dung lượng tính bằng bit	Phạm vi
signed short int	16	Giống như kiểu short int
long int	32	-2,147,483,648 tới 2,147,483,647
signed long int	32	0 tới 4,294,967,295
unsigned long int	32	Giống như kiểu long int
float	32	6 con số thập phân
double	64	10 con số thập phân
long double	128	10 con số thập phân

Tên và hằng trong C

- Tên (danh biểu): Tên hay còn gọi là danh biểu (identifier) được dùng để đặt cho chương trình, hằng, kiểu, biến, chương trình con... Tên có hai loại là tên chuẩn và tên do người lập trình đặt.
- 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 để dùng trong chương trình của mình. Sử dụng bộ chữ cái, chữ số và dấu gạch dưới (_) để đặt tên, nhưng phải tuân thủ quy tắc:

Quy tắc đặt tên

- 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 không giới hạn, tuy nhiên chỉ có 31 ký tự đầu tiên là có ý nghĩa.
- Không cấm việc đặt tên trùng với tên chuẩn nhưng khi đó ý nghĩa của tên chuẩn không còn giá trị nữa.
- *Ví dụ*: tên do người lập trình đặt: Chieu_dai, Chieu_Rong, Chu_Vi, Dien_Tich
- Tên không hợp lệ: Do Dai, 12A2,…

Hằng

- Một **hằng** (constant) là một giá trị không bao giờ thay đổi trong thời gian tồn tại của nó.
- Định nghĩa hằng: sử dụng từ khóa **const**

  const <kiểu dữ liệu> <tên hằng> = <giá trị>

Hằng

Các ví dụ

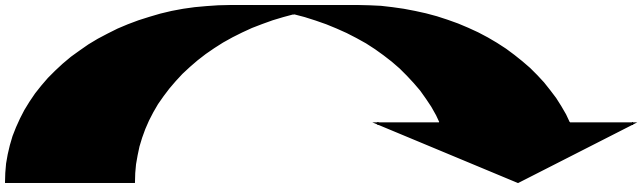
- const int a= 5; **hằng số nguyên**
- const float x = 5.3; **hằng số thực**
- const char c = ‘1’; **hằng ký tự**

- Hằng trong hệ 16 được bắt đầu bằng 0x.
  Ví dụ: 0xa5 = 10*16 + 5 =165.
- Hằng trong hệ 8 bắt đầu bằng 0.
  Ví dụ: 0345 = 3*64+4*16+5=229

Biến và biểu thức

- Biến là một đại lượng được người lập trình định nghĩa và được đặt tên thông qua việc khai báo biến.
- Biến dùng để chứa dữ liệu trong quá trình thực hiện chương trình và giá trị của biến có thể bị thay đổi trong quá trình này.
- Cách đặt tên biến giống như cách đặt tên đã nói trong phần trên.
- Mỗi biến thuộc về một kiểu dữ liệu xác định và có giá trị thuộc kiểu đó.

Biến



Bộ nhớ

Dữ liệu 15

	15		
	Dữ liệu trong bộ nhớ		

Mỗi vị trí trong bộ nhớ là duy nhất

Biến cho phép cung cấp một tên có ý nghĩa cho mỗi vị trí nhớ

Khai báo biến

- **<kiểu dữ liệu> <tên biến> [=<giá trị 1>]**

- **Ví dụ:**

```
int a = 3;
int b;
int a=3, b=4;
char c = ‘A’;
```

Ví dụ về cách khai báo biến

```
#include stdio.h
#include math.h
Int template;       /* bien toan cuc*/
main ()
   {
      char abc;       /*bien cuc bo */
   ……….
   }
```

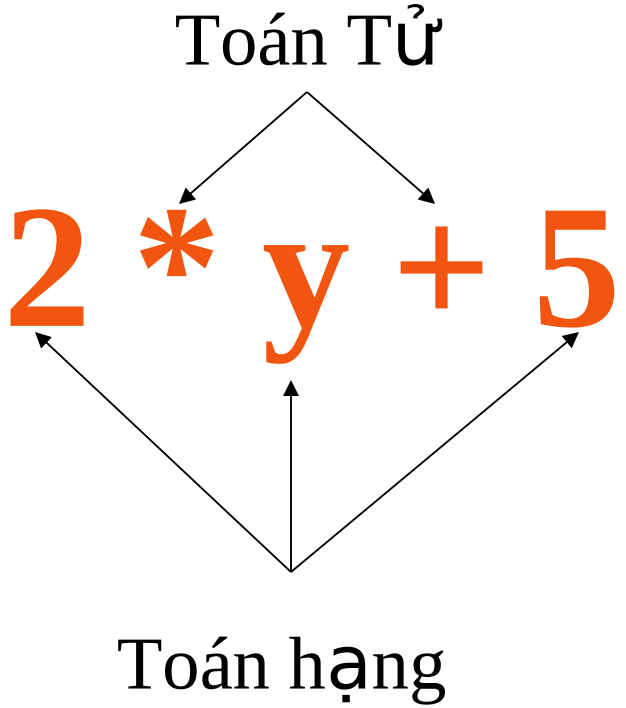
Biểu thức trong C

- Biểu thức là một sự kết hợp giữa các toán tử (operator) và các toán hạng (operand) theo đúng một trật tự nhất định.
- Mỗi toán hạng có thể là một hằng, một biến hoặc một biểu thức khác.
- Trong trường hợp, biểu thức có nhiều toán tử, ta dùng cặp dấu ngoặc đơn () để chỉ định toán tử nào được thực hiện trước

Biểu thức (Expressions)

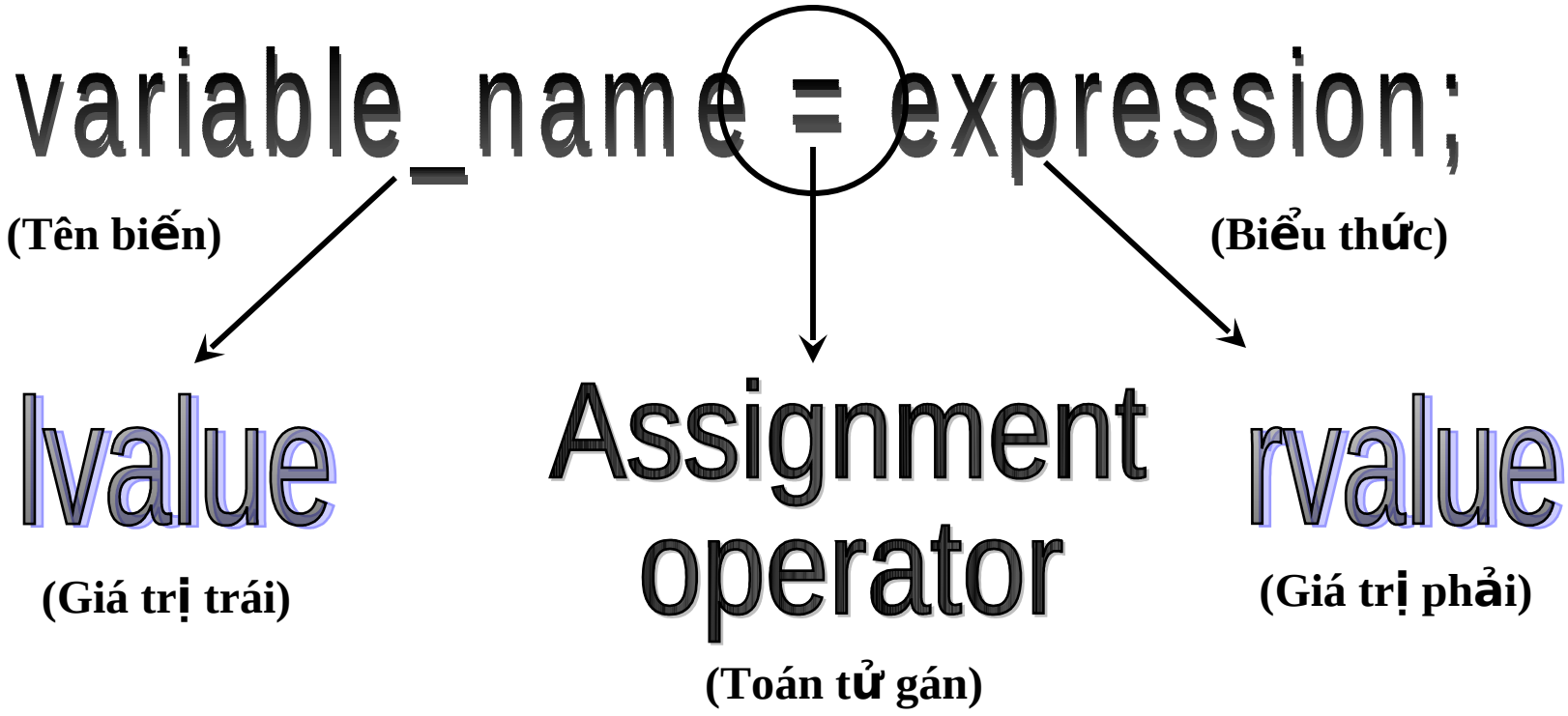
Sự kết hợp các toán tử và các toán hạng

Ví dụ:



Toán tử gán

Toán tử gán (=) có thể được dùng với bất kỳ biểu thức C hợp lệ nào



Gán liên tiếp

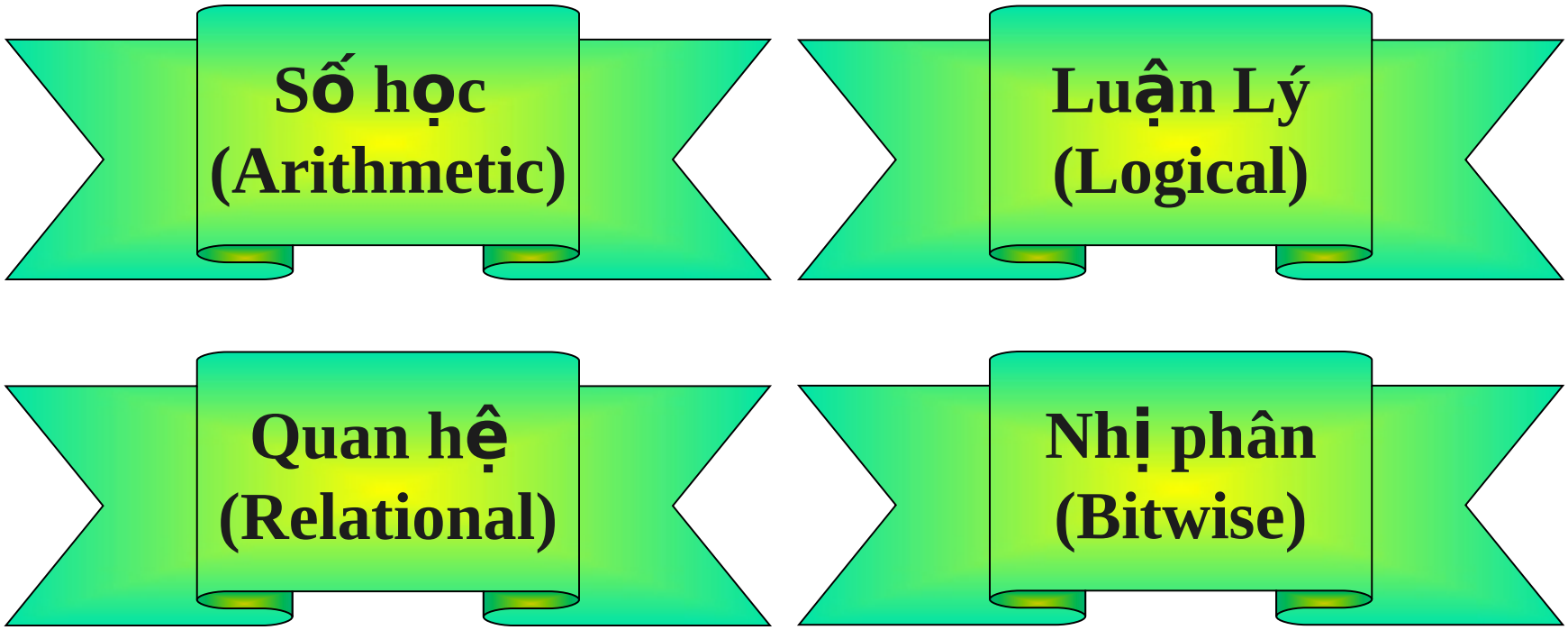
Nhiều biến có thể được gán với cùng một giá trị trong một câu lệnh đơn

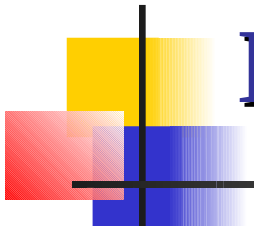
```
a = b = c = 10;
```

Tuy nhiên, không thể áp dụng quy tắc trên khi khai báo biến

```
int a = int b = int b = int c = 10
```

Bốn Kiểu Toán Tử





Biểu thức số học

Biểu thức số học có thể được biểu diễn trong C bằng cách sử dụng các toán tử số học

Ví dụ :

```
++i % 7

5 + (c = 3 + 8)

a * (b + c/d) - 22
```

Toán tử số học

Các phép toán hai ngôi số học là

Phép toán	Ý nghĩa	Ví dụ
+	Phép cộng	a+b
-	Phép trừ	a-b
*	Phép nhân	a*b
/	Phép chia	a/b (Chia số nguyên sẽ chặt phần thập phân)
%	Phép lấy phần dư	a%b (Cho phần dư của phép chia a cho b)

Có phép toán một ngôi - ví du -(a+b) sẽ đảo giá trị của phép cộng (a+b).

Toán tử quan hệ và luận lý

Được dùng để :

Kiểm tra mối quan hệ giữa hai biến hay giữa một biến và một hằng

Toán tử quan hệ

Toán tử	Ý nghĩa
>	Lớn hơn
>=	Lớn hơn hoặc bằng
<	Nhỏ hơn
<=	Nhỏ hơn hoặc bằng
==	Bằng
!=	Không bằng

Toán tử quan hệ và luận lý (tt.)

Toán tử luận lý là những ký hiệu dùng để kết hợp hay phủ định biểu thức chứa các toán tử quan hệ

Toán tử	**Ý nghĩa**
&&	**AND**: Kết quả là True khi cả 2 điều kiện đều đúng
||	**OR** : Kết quả là True khi chỉ một trong hai điều kiện là đúng
!	**NOT**: Tác động trên các giá trị riêng lẻ, chuyển đổi True thành False và ngược lại.

Ví dụ: if (a>10) && (a<20)

Những biểu thức dùng toán tử luận lý trả về 0 thay cho false và 1 thay cho true

Toán tử luận lý nhị phân

Dữ liệu chỉ được xử lý sau khi đã chuyển đổi giá trị SỐ thành giá trị NHỊ PHÂN

Toán tử	**Mô tả**
Bitwise AND (x & y)	Mỗi vị trí của bit trả về kết quả là 1 nếu bit của hai toán hạng là 1.
Bitwise OR (x | y)	Mỗi vị trí của bit trả về kết quả là 1 nếu bit của một trong hai toán hạng là 1.
Bitwise NOT (~ x)	Đảo ngược giá trị của toán hạng (1 thành 0 và ngược lại).
Bitwise XOR (x ^ y)	Mỗi vị trí của bit chỉ trả về kết quả là 1 nếu bit của một trong hai toán hạng là 1 mà không không phải cả hai toán hạng cùng là 1.

Toán tử luận lý nhị phân (tt.)

Ví dụ

- 10 & 15→ **1010 & 1111→1010** → 10
- 10 | 15→ **1010 | 1111→1111** → 15
- 10 ^ 15→ **1010 ^ 1111→0101** → 5
- ~ **10** → ~**1010** →1…11110**101** → **-11**

Chuyển đổi kiểu

Qui tắc chuyển đổi kiểu tự động trình bày dưới đây nhằm xác định giá trị biểu thức:

a. char và short được chuyển thành int và float được chuyển thành double.

b. Nếu có một toán hạng là double, toán hạng còn lại sẽ được chuyển thành double, và kết quả là double.

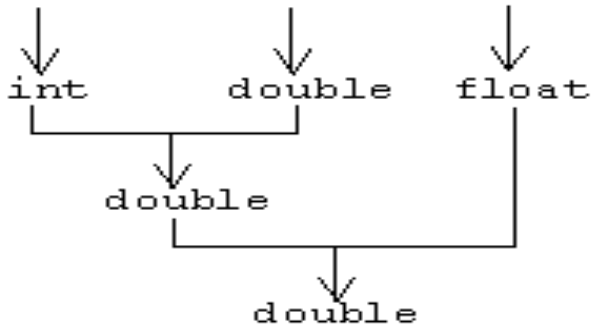
c. Nếu có một toán hạng là long, toán hạng còn lại sẽ được chuyển thành long, và kết quả là long.

d. Nếu có một toán hạng là unsigned, toán hạng còn lại sẽ được chuyển thành unsigned và kết quả cũng là unsigned.

e. Nếu tất cả toán hạng kiểu int, kết quả là int.

Ví dụ

```
char ch;
int i;
float f;
double d;
result = (ch/i) + (f*d) - (f+i);
```



Ép kiểu

Một biểu thức được ép thành một kiểu nhất định bằng cách dùng kỹ thuật ép kiểu (**cast).**

Cú pháp :

(kiểu dữ liệu) cast

Kiểu → Bất cứ kiểu dữ liệu hợp lệ trong C

Ví dụ:

```
float x,f;
f = 3.14159;
x = (int) f;
```

Giá trị số nguyên trả về bởi (int) f được chuyển thành số thực khi nó được toán tử GÁN xử lý. Song, giá trị của f vẫn không đổi.

Giá trị của x sẽ là 3 (số nguyên)

Độ ưu tiên của toán tử

- Độ ưu tiên tạo nên cấu trúc phân cấp của loại toán tử này so với loại toán tử khác khi tính giá trị một biểu thức số học
  - Nó đề cập đến thứ tự thực thi các toán tử trong C
  - Độ ưu tiên của các toán tử này được thay đổi bởi các dấu ngoặc đơn trong biểu thức

Loại toán tử	Toán tử	Tính kết hợp
Một ngôi	- ++ --	Phải đến trái
Hai ngôi	^	Trái đến phải
Hai ngôi	* / %	Trái đến phải
Hai ngôi	+ -	Trái đến phải
Hai ngôi	=	Phải đến trái

Độ ưu tiên của toán tử (tt.)

Ví dụ: -8 * 4 % 2 - 3

Trình tự	Thao tác	Kết quả
1.	- 8 (phép trừ một ngôi)	số âm của 8
2.	- 8 * 4	- 32
3.	- 32 % 2	16
4.	16-3	13

Độ ưu tiên của toán tử so sánh

Độ ưu tiên của toán tử so sánh (quan hệ) luôn được tính từ trái sang phải

Độ ưu tiên của toán tử luận lý

Thứ tự ưu tiên	Toán tử
1	NOT
2	AND
3	OR

Khi có nhiều toán tử luận lý trong một điều kiện, ta áp dụng quy tắc tính từ phải sang trái

Độ ưu tiên của toán tử luận lý (tt.)

Xét biểu thức sau:

False OR True AND NOT False AND True

Điều kiện này được tính như sau:

False OR True AND [NOT False] AND True

NOT có độ ưu tiên cao nhất.

False OR True AND [True AND True]

Ở đây, AND có độ ưu tiên cao nhất, những toán tử có cùng ưu tiên được tính từ phải sang trái.

False OR [True AND True]

[False OR True]

True

Độ ưu tiên giữa các toán tử

Khi một biểu thức có nhiều loại toán tử thì độ ưu tiên giữa chúng phải được thiết lập.

Thứ tự ưu tiên	Kiểu toán tử
1	Số học (Arithmetic)
2	So sánh (Comparison)
3	Luận lý (Logical)

Độ ưu tiên giữa các toán tử (tt.)

Ví dụ :

2*3+4/2 > 3 AND 3<5 OR 10<9

Việc tính toán như sau :

[2*3+4/2] > 3 AND 3<5 OR 10<9

Toán tử số học sẽ được tính trước

[[2*3]+[4/2]] > 3 AND 3<5 OR 10<9

[6+2] >3 AND 3<5 OR 10<9

[8 >3] AND [3<5] OR [10<9]

Độ ưu tiên giữa các toán tử (tt.)

Kế đến là toán tử so sánh có cùng độ ưu tiên. Ta áp dụng quy tắc tính từ trái sang phải.

True AND True OR False

Cuối cùng là toán tử kiểu luận lý. AND sẽ có độ ưu tiên cao hơn OR

[True AND True] OR False

True OR False

True

Thay đổi độ ưu tiên

- Dấu ngoặc đơn ( ) có độ ưu tiên cao nhất
- Độ ưu tiên của các toán tử có thể được thay đổi bởi dấu ngoặc đơn
- Toán tử có độ ưu tiên thấp hơn nếu đặt trong dấu ngoặc đơn sẽ được thực thi trước
- Khi các cặp ngoặc đơn lồng nhau ((( ))), cặp ngoặc đơn trong cùng nhất sẽ được thực thi trước
- Nếu trong biểu thức có nhiều cặp ngoặc đơn thì việc thực thi sẽ theo thứ tự từ trái sang phải

Thay đổi độ ưu tiên (tt.)

Ví dụ :

5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR (2<6 AND 10>11))

Cách tính :

1) **5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR (True AND False))**

Dấu ngoặc đơn bên trong sẽ được tính trước

2) **5+9*3^2-4 > 10 AND (2+2^4-8/4 > 6 OR False)**

Thay đổi độ ưu tiên (tt.)

3) 5+9*3^2-4 >10 AND (2+16-8/4 > 6 OR False)

Kế đến dấu ngoặc đơn ở ngoài được tính đến

4) 5+9*3^2-4 > 10 AND (2+16-2 > 6 OR False)

5) 5+9*3^2-4 > 10 AND (18-2 > 6 OR False)

6) 5+9*3^2-4 > 10 AND (16 > 6 OR False)

7) 5+9*3^2-4 > 10 AND (True OR False)

8) 5+9*3^2-4 > 10 AND True

Thay đổi độ ưu tiên (tt.)

9) 5+9*9-4>10 AND True

Biểu thức bên trái được tính trước

10) 5+81-4>10 AND True

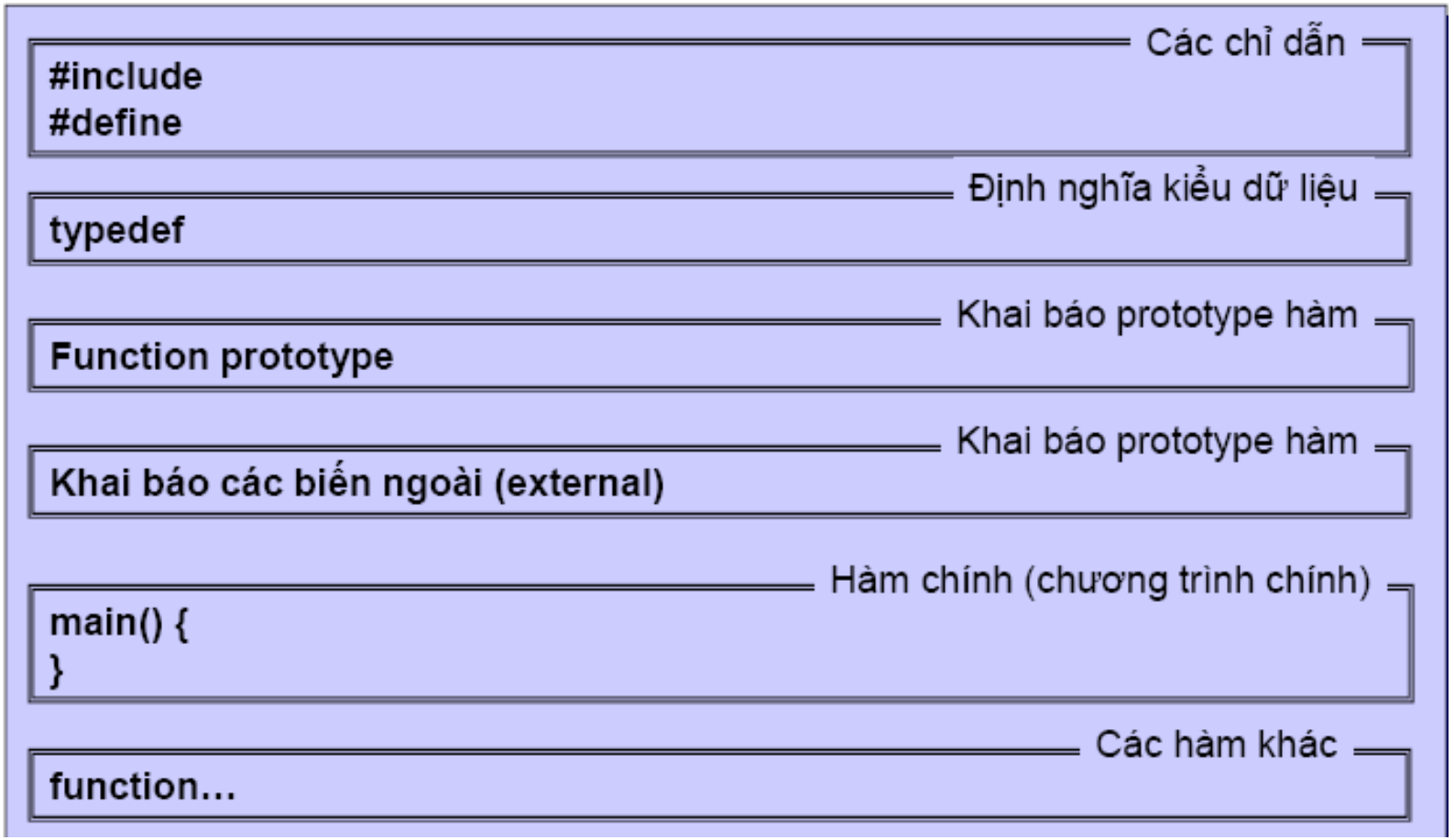
11) 86-4>10 AND True

12) 82>10 AND True

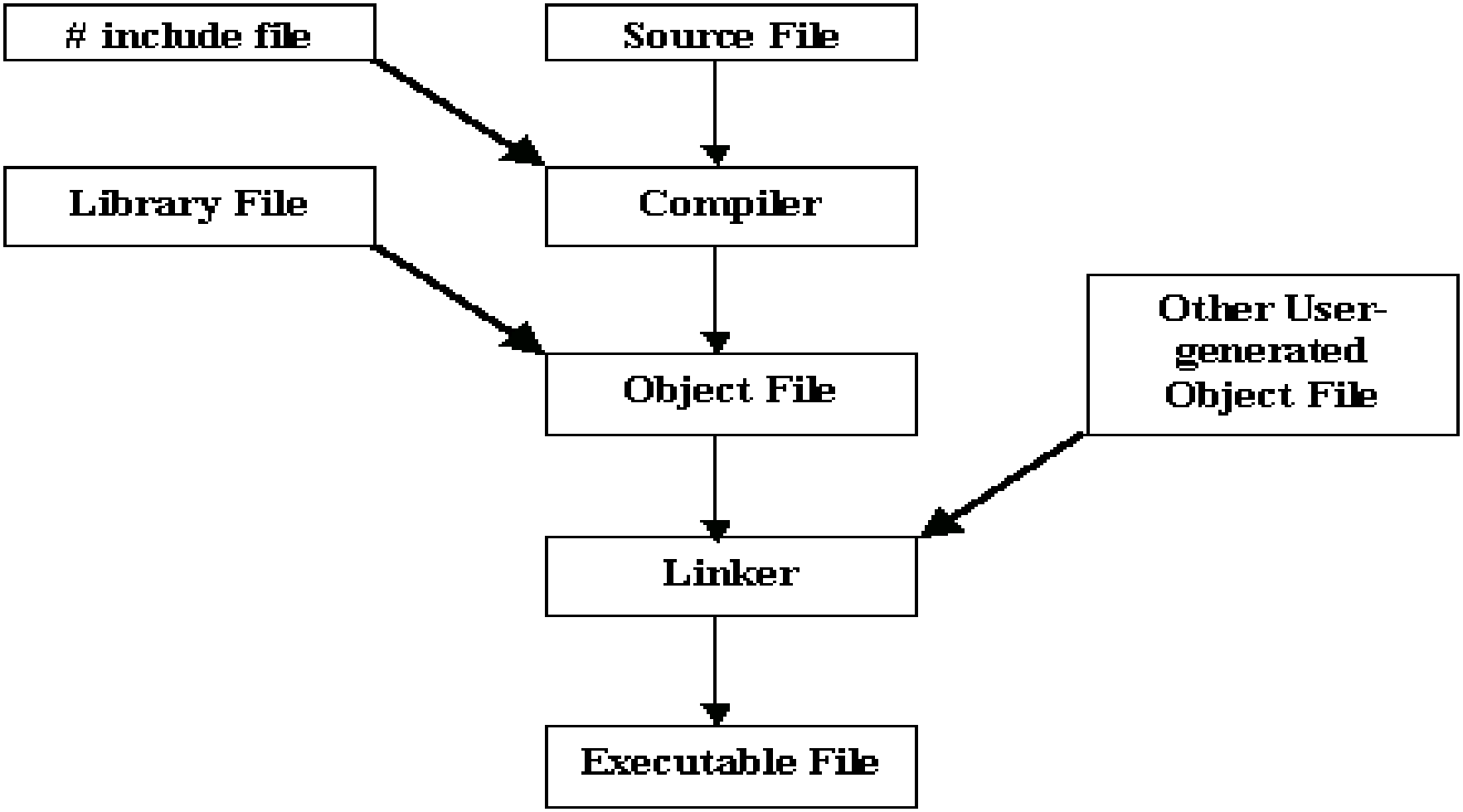
13) True AND True

14) True

Cấu trúc của một chương trình trong C



Biên dịch và thi hành chương trình



Một số ví dụ đơn giản

```
#include "stdio.h“ //standard io
#include "conio.h“ //console io
void main()
{
    clrscr();
    printf("Hello World!");
    getch();
}
```

hello.c

```
#include "stdio.h"
#include "conio.h"
void main()
{
    int a,b,c;
    clrscr();
    printf("Nhap vao hai so nguyen:\n");
    scanf("%d%d",&a,&b);
    c=a+b;
    printf("Tong hai so la: %d",c);
    getch();
}
```

add2num.c

Ví dụ chương trình C đơn giản

```c
#include <stdio.h>

int main(int argc, char *argv[])
{
    /* Xuat chuoi ra man hinh */
    printf("Hello C-Free!\n");

    /* In tong cua hai so ra man hinh */
    printf("Total: %d\n",total(10, 20));
    return 0;
}
/* Ham tinh tong cua hai so */
int total(int a, int b){
    return (a+b);
}
```