Chương I

Tổng quan về ngôn ngữ lập trình C

1.1 Giới thiệu chung

C và ngôn ngữ phát triển của nó là C++ được phổ biến khá rộng rãi và là một trong những ngôn ngữ lập trình chủ yếu trong việc xây dựng những phần mềm hiện nay.

Lịch sử phát triển :

1972

- C được ra đời và phát triển bởi hai nhà khoa học máy tính là ***Brian W.Kernighan*** và ***Dennis Ritchie.***

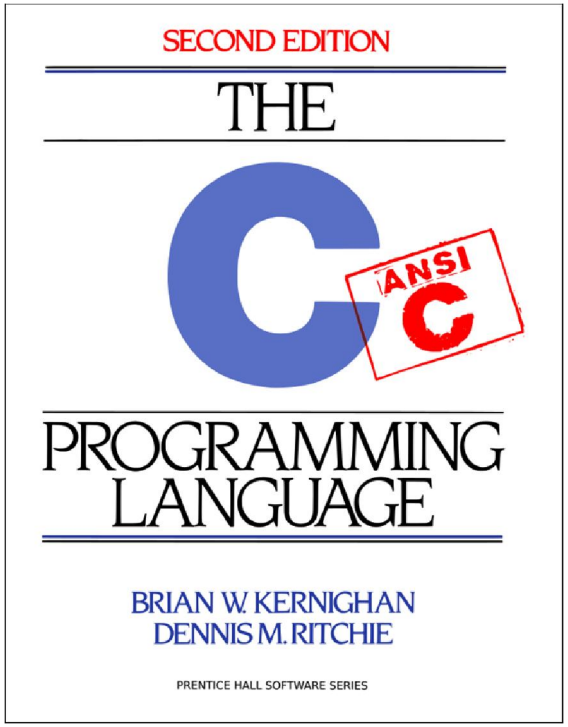
1978

- Cuốn sách "*The C Programming Language*" được xuất bản lần đầu tiên để giới thiệu ngôn ngữ C.

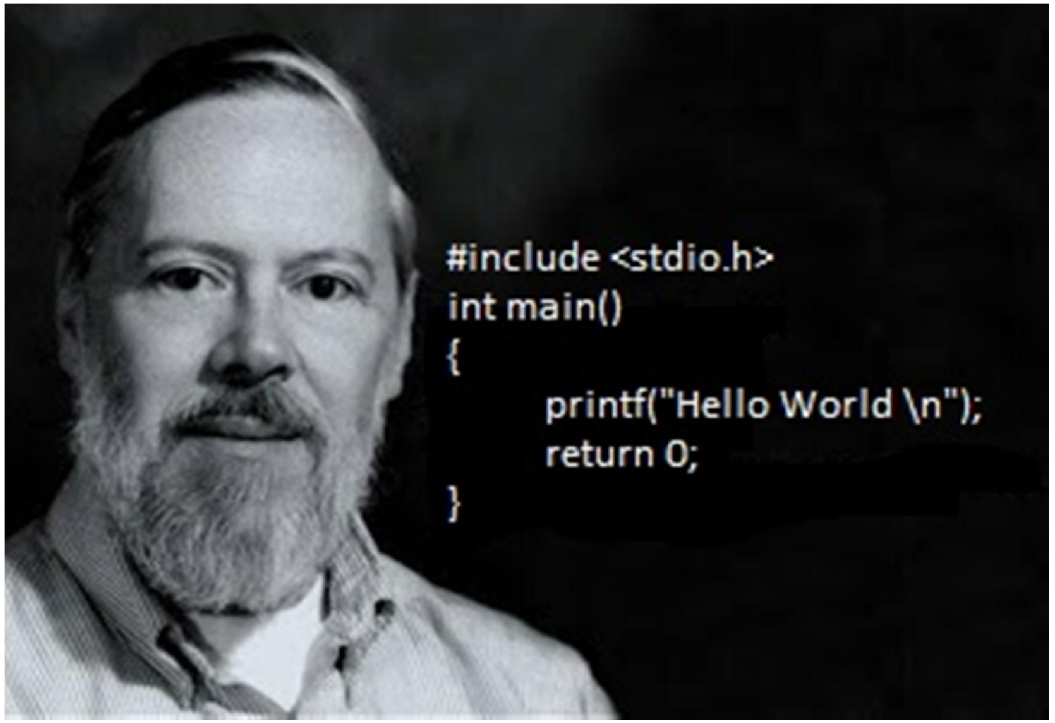
1989

- Phiên bản chuẩn hóa ANSI được công bố trong cuốn "*The C Programming Language*". Xuất bản lần hai

1.1 Giới thiệu chung



Cuốn "The c programming language" xuất bản lần 2



Dennis MacAlistair Ritchie
(09/09/1941 – 12/10/2011)

Đặc điểm của ngôn ngữ C

- *Phân biệt chữ hoa và chữ thường.*
- *Có số phép toán và thư viện hàm phong phú.*
- *Các biểu thức được biểu diễn bằng những chuỗi ký tự ngắn gọn*
- *Tương thích với nhiều hệ điều hành như Unix, Windows…*

Trình biên dịch

Trình biên dịch hay phần mềm biên dịch ***(compiler)*** là một chương trình máy tính làm công việc dịch một chuỗi các câu lệnh được viết bằng một ngôn ngữ lập trình thành một chương trình tương đương nhưng ở dưới dạng một ngôn ngữ máy tính.

Những trình dịch về C ngày nay thường được cung cấp kèm chung với C++. Sau đây là danh sách một số trình dịch phổ biến:

- *GCC*
- *Borland C/C++*
- *Microsoft Visual Studio*
- *Turbo C/C++*
- *C Free*
- *Dev C/C++*
- *Code Block*

1.2 Bộ kí tự và từ khóa

Bộ chữ viết trong ngôn ngữ C bao gồm những kí tự, ký hiệu 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 (Trình biên dịch sẽ bỏ qua kí tự khoảng trắng (space) nếu nó không nằm trong một hằng chuỗi.)*

1.2 Bộ kí tự và từ khóa

Từ khóa là các từ dành riêng (reserved words) của một ngôn ngữ mà người lập trình. Mỗi từ khóa có một ý nghĩa xác định và chúng ta không thể thay đổi nó.

Dưới đây là bộ từ khóa của ngôn ngữ C:

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

1.3 Định danh (đặt tên)

Định danh là một dãy kí tự dùng để gọi tên các đối tượng trong chương trình như biến, hằng, hàm, mảng,…

Một số qui tắc cần tuân theo khi đặt tên trong C:

- Không được bắt đầu bằng chữ số, không được trùng với từ khóa.
- Chỉ được sử dụng các ký tự gồm chữ cái (A..Z,a..z), chữ số (0..9) và dấu gạch dưới '_'.

Ví dụ: dien_tich /*Định danh hợp lệ*/

dien tich /*Định danh không hợp lệ*/

1.4 Các kiểu dữ liệu chuẩn

Kiểu		Kích thước	Miền giá trị
Kí tự	char	1 byte	-128 → +127
	unsigned char	1 byte	0 → 255
Số nguyên	int	2 byte	-32768 → 32767 (- 2^{15} →2^{15}-1)
	unsigned int	2 byte	0 → 65535 (0→2^{16} - 1)
	long	4 byte	-2147483648 → 2147483647 (-2^{31}→2^{31}-1)
	unsigned long	4 byte	0 → 4294967295 (0→ 2^{32} -1)
Số thực	float	4 byte	$3.4*10^{-38}$ → $3.4*10^{38}$
	double	8 byte	$1.7*10^{-308}$ → $1.7*10^{308}$
	long double	10 byte	$3.4*10^{4932}$ → $1.1*10^{4932}$

1.4 Các kiểu dữ liệu chuẩn

Chú ý:

- *Kiểu ký tự cũng có thể xem là một dạng của kiểu số nguyên.*
- *Ngoài kiểu kí tự, kiểu số nguyên và số thực ra, trong C còn có kiểu dữ liệu **void**, kiểu này mang ý nghĩa là kiểu rỗng không chứa giá trị gì cả.*

1.5 Biến

- 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 giá trị thuộc một kiểu dữ liệu xác định trong quá trình thực hiện chương trình.
- Giá trị của biến có thể bị thay đổi nhưng kiểu dữ liệu của nó thì không.

Khai báo biến

Biến phải được khai báo trước khi sử dụng. Tùy trường hợp mà có thể lựa chọn các cách khai báo biến sau:

- Cú pháp khai báo chung:

kiểu_dữ _liệu tên_biến ;

- Khai báo nhiều biến có cùng một kiểu dữ liệu:

kiểu_dữ _liệu tên_biến1, tên_biến2,…;

- Khai báo và khởi tạo giá trị cho biến:

kiểu_dữ _liệu tên_biến = giá trị_khởi_tạo ;

Vị trí khai báo biến

- Khai báo bên ngoài các khối lệnh: (Biến ngoài)

  – Phạm vi sử dụng: từ vị trí khai báo xuống các khối lệnh bên dưới.

  – Giá trị ban đầu: bằng 0.

  – Thời gian tồn tại: cho đến khi kết thúc chương trình.

- Khai báo bên trong khối lệnh: (Biến trong)

  – Phạm vi sử dụng: bên trong khối lệnh đó và cả các khối lệnh lồng bên trong khối đó.

  – Giá trị ban đầu: chưa được xác định

  – Thời gian tồn tại: Khi thực hiện xong khối lệnh

1.6 Hằng

Hằng (constant) - là đại lượng không đổi trong suốt quá trình thực thi của chương trình.

Hằng có thể là một chuỗi ký tự, một ký tự, một con số xác định. Để đặt tên một hằng, ta dùng dòng lệnh sau :

#define Tên_hằng Giá_trị

Hoặc

const Kiểu_dữ_liệu Tên_hằng = Giá_trị ;

Ví dụ: #define PI 3.14

const int MAX = 100;

Hằng số nguyên

- Dạng thập phân: *Giá trị viết ở dạng số nguyên thập phân.*

  Ví dụ: ***const int x = 100 ;***

- Dạng bát phân: *Giá trị nguyên bát phân được viết sau số 0.*

  Ví dụ: ***#define x 0144*** /*Hằng x có giá trị nguyên bát phân bằng 144*/

- Dạng thập lục phân: *Giá trị nguyên thập lục phân viết sau 0x hoặc 0X.*

  Ví dụ: ***#define x 0x64*** /*Hằng x có giá trị ở hệ thập lục phân bằng 64*/

Lưu ý: *Để biểu diễn các hằng kiểu **long**, **unsigned int**, hoặc **unsigned long** người ta thường thêm hậu tố L hoặc l (long), U hoặc u (unsigned int), UL hoặc ul (unsigned long) vào cuối giá trị nguyên.*

Hằng số thực

Hằng số thực được thể hiện theo 2 cách sau:

- Sử dụng cách viết thông thường (dấu phẩy tĩnh), cần lưu ý là sử dụng dấu thập phân là dấu chấm.

  Ví dụ: ***const float Pi = 3.14 ;***

- Sử dụng cách viết theo số mũ hay số khoa học (dấu phẩy động). Một số thực được tách làm 2 phần, cách nhau bằng ký tự e hay E.

  Ví dụ: ***#define x 12.3e-3*** //x= $12.3*10^{-3} = 0.0123$

Chú ý: - *Thêm hậu tố cho kiểu **double** là **F***

- *Thêm hậu tố cho kiểu **long double** là **L***

Ví dụ: ***#define Pi = 3.14L ;***

Hằng kí tự

Hằng ký tự là một ký tự riêng biệt được viết trong cặp dấu nháy đơn . Mỗi một ký tự tương ứng với một giá trị trong bảng mã ASCII. Hằng ký tự cũng được xem như trị số nguyên.

Ví dụ: ‘a’, ‘A’, ‘0’, ‘9’

Chúng ta có thể thực hiện các phép toán số học trên 2 kí tự (thực chất là thực hiện phép toán trên giá trị ASCII của chúng)

Hằng chuỗi kí tự

Hằng chuỗi ký tự là một chuỗi hay một xâu ký tự được đặt trong cặp dấu nháy kép .

Ví dụ: *"Ngon ngu lap trinh C"*

Chú ý:

- 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: "I'm a student" phải viết "I\'m a student"

"Day la ky tu "dac biet"" phải viết "Day la ky tu \"dac biet\""

1.7 Biểu thức và toán tử

Biểu thức là một chuỗi gồm các toán hạng và toán tử được kết hợp với nhau.

Mỗi toán hạng có thể là hằng, biến, lời gọi hàm, hoặc biểu thức con.

Mỗi biểu thức sẽ có một giá trị xác định. Giá trị đó có thể là giá trị số học hoặc giá trị logic: true (1), false (0).

Toán tử số học

Bao gồm các phép toán:

Cộng	Trừ	Nhân	Chia	Lấy dư
+	-	*	/	%

Ví dụ : 9%4 = 1 (9 chia 4 dư 1).

–7 + 2 * ((4 + 3) * 4 + 8) = 65.

Lưu ý: *Toán tử % chỉ áp dụng cho kiểu số nguyên. Phép chia giữa hai giá trị nguyên sẽ cho kết quả là giá trị nguyên (Ví dụ: 3/4 = 0).*

Toán tử quan hệ (so sánh)

>	*Lớn hơn*	<=	*Nhỏ hơn hoặc bằng*
>=	*Lớn hơn hoặc bằng*	==	*So sánh bằng*
<	*Nhỏ hơn*	!=	*So sánh khác*

Kết quả của phép toán quan hệ là số nguyên kiểu int, bằng 1 nếu đúng, bằng 0 nếu sai.

Chú ý: *Trong các phép toán, toán tử số học* ***ưu tiên trước*** *toán tử quan hệ .*

Ví dụ: 3 == 5 = 0 (sai)

6 – 3 < 4 = 1 (đúng), tương đương (6 – 3) < 4

–2 * –4 < 3 + 2 = 0 (sai), tức là (–2 * –4) < (3 + 2)

Toán tử logic

NOT (phép phủ định)	AND (phép và)	OR (phép hoặc)
!	**&&**	**||**

Thứ tự ưu tiên :

! →Toán tử số học →Toán tử quan hệ →&&→ ||

Bảng giá trị :

Toán hạng a	**Toán hạng b**	**!a**	**a&&b**	**a||b**
Khác 0	Khác 0	0	1	1
Khác 0	0	0	0	1
0	Khác 0	1	0	1
0	0	1	0	0

Toán tử xử lý bit

&	*Và (AND)*	>>	*Dịch phải (RoR)*
|	*Hoặc (OR)*	<<	*Dịch trái (RoL)*
^	*Hoặc loại trừ (XOR)*	~	*Đảo (NOT)*

Bảng giá trị :

bit a	bit b	~a	a&b	a|b	a^b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

Ví dụ

a = 77 → đổi ra hệ nhị phân → 0000 0000 0100 1101
b = 29 → đổi ra hệ nhị phân → 0000 0000 0001 1101 *(lấy 2 Byte)*

	0000 0000 0100 1101	0000 0000 0100 1101
	0000 0000 0001 1101	0000 0000 0001 1101
a&b =	0000 0000 0000 1101	a|b = 0000 0000 0101 1101
=	13 (dạng thập phân)	= 93(dạng thập phân)

~a = 1111 1111 1011 0010 = -78 (dạng thập phân)

a>>2 = 0000 0000 0001 0011 = 19 (dạng thập phân)

a<<3 = 0000 0000 0110 1000 = 616 (dạng thập phân)

Lưu ý: a<<N = a*2^N

a>>N = a/2^N (kết quả lấy phần nguyên)

Toán tử gán

Toán tử gán dùng nhằm thay thế giá trị hiện tại của biến bằng một giá trị mới.

Các phép gán bao gồm:

=, +=, – =, *=, /=, %=, <<=, >>=, &=, |=, ^=.

Ví dụ : Ta có giá trị i = 3

i = i + 3 → i = 6

i += 3 → i = 6 ⇔ i = i + 3

i *= 3 → i = 9 ⇔ i = i * 3

Toán tử tăng giảm

- Phép toán tăng ++ sẽ cộng thêm 1.

  Ví dụ: ++n; hay n++; ⇔ n = n+1;

- Phép toán giảm -- sẽ trừ đi 1.

  Ví dụ: --n; hay n--; ⇔ n = n -1;

Lưu ý: *Trường hợp sử dụng toán tử này trong một biểu thức thì việc đặt trước hay sau sẽ ảnh hưởng đến kết quả bài toán:*

- Đặt trước: Để thay đổi giá trị cho n trước khi sử dụng n.

- Đặt sau: Để thay đổi giá trị cho n sau khi sử dụng n xong.

Ví dụ: Với n = 4.

Lệnh x = ++n; ⇔ n = n + 1; x = n;

Lệnh x = n++; ⇔ x = n; n = n + 1;

Toán tử dấu phẩy

(biểu _thức_1, biểu_thức_2,…, biểu_thức_n)

- Thực hiện từ trái sang phải.
- Kết quả và kiểu dữ liệu của biểu thức dấu phẩy là của biểu thức cuối cùng <**biểu_thức_n**>.

Ví dụ: m = (t = 2, t*t + 3); ⇔ m = 7

Toán tử điều kiện 3 ngôi "?"và ":"

Điều_kiện? biểu_thức_1:biểu_thức_2

Giá trị sẽ là ***biểu_thức_1*** nếu ***Điều_kiện*** có giá trị đúng, ngược lại giá trị sẽ là ***biểu_thức_2.***

Ví dụ: Với x = 2 thì

t = (x >= 0 ? x : x*-1); ⇔ t = 2

Chuyển kiểu dữ liệu

(Tên_kiểu) Biểu_thức

- Phép chuyển kiểu cho ra giá trị thuộc kiểu chỉ định.
- Bản thân của biểu thức thì không thay đổi kiểu.

Ví dụ 1: Đổi số thực sang số nguyên.

```
int a;
float b = 2.3;
a = (int)( b + 0.5); //Kết quả a=2
```

Ví dụ 2:

```
int a = 5, b = 2;
float x1, x2;
x1 = (float) a / b; //Kết quả x1=2.5
x2 = a / b; //Kết quả x2 = 2
```

1.8 Độ ưu tiên giữa các toán tử

Độ ưu tiên	Toán tử	Trình tự kết hợp
1	() [] -> .	→
2	! ~ ++ -- - + * (type) &	←
3	sizeof	→
4	* / %	→
5	+ -	→
6	<< >>	→
7	< <= > >=	→
8	== !=	→
9	&	→
10	^	→
11	|	→
12	&&	→
13	||	←
14	?:	←
15	= += -= *= /= %= &= …	←

1.9 Cấu trúc cơ bản của một chương trình c

- Các chỉ thị **#include** (khai báo tiền xử lý): Dùng nạp file chứa các hàm thư viện sử dụng trong chương trình.
- Các chỉ thị **#define**: Dùng định nghĩa hằng, hàm, kiểu dữ liệu (nếu cần).
- Khai báo các đối tượng dữ liệu bên ngoài hàm:
  - Biến.
  - Khai báo nguyên mẫu hàm.
  - Kiểu dữ liệu mới ...
- Hàm **main**: Chứa các lệnh cần thực hiện tuần tự từ trên xuống.
- Định nghĩa hàm (đã được khai báo tiền xử lý).

Lệnh tiền xử lý *#include*

Đưa ra chỉ thị cho trình biên dịch thực hiện liên kết đến tệp tin thư viện có chứa một số hàm mà chương trình cần sử dụng.

Cách dùng : ***#include<tên_tệp.h>***

Ví dụ :

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

Câu lệnh và khối lệnh

- **Câu lệnh** là một chỉ thị nhằm ra lệnh cho chương trình thực hiện một tác vụ cụ thể nào đó. Mỗi câu lệnh có thể được viết trên một hoặc nhiều dòng, và được kết thúc bằng ***dấu chấm phẩy***.

  Câu lệnh được phân chia thành 2 loại:

  - Câu lệnh đơn: *là câu lệnh không chứa câu lệnh khác: câu lệnh gán, lệnh khai báo, lệnh xuất nhập...*

  - Câu lệnh phức: *là câu lệnh có chứa câu lệnh khác bên trong nó như khối lệnh, câu lệnh rẽ nhánh, câu lệnh lặp, ...*

- **Khối lệnh** gồm một hoặc nhiều câu lệnh đơn được bao bởi cặp dấu ngoặc {}. Một khối lệnh có thể lồng bên trong nó một hoặc nhiều khối lệnh khác.

Chú thích trong C

Khi viết chương trình đôi lúc ta cần phải có vài lời ghi chú về một đoạn chương trình nào đó để dễ nhớ và dễ điều chỉnh sau này. Trong ngôn ngữ lập trình C, nội dung chú thích có thể được viết bằng hai cách:

- Cách 1:

```
/*chú_thích*/
```

Cách này có thể viết chú thích trên một hoặc nhiều dòng.

- Cách 2:

```
//chú_thích
```

Cách này chỉ viết chú thích trên một dòng (tức là chú thích kết thúc khi ta ấn phím enter).

Chú ý: *Chú thích có thể được viết ở bất kì vị trị nào trong chương trình và nó không ảnh hưởng gì đến kết quả chạy chương trình.*

Bài tập luyện tập

Bài 1: Các định danh nào sau là hợp lệ ? tại sao?

a) Xx	f) Phieu.du
b) _ABC	g) _123
c) Ban Kinh	h) A$
d) char	i) _
e) 999_doa_hong	j) 1_2_3

Bài tập luyện tập

Bài 2: Những biểu tượng nào sau đây là hằng ? Nếu là hằng thì nó thuộc kiểu dữ liệu nào ?

a) 123.456	j) 123.5e2	s) 1234uL
b) 0x10.5	k) .0001	t) 1.2Fe-7
c) 0X0G1	l) +12	u) 15,000
d) 0001	m) 98.6F	v) 1.234L
e) 0xFFFF	n) 98.7U	w) 197u
f) 123L	o) “Tin hoc”	x) 100U
g) 0Xab05	p) 0996	y) 0XABCDEFL
h) 0L	q) -12E-12	z) 0xabcu
i) -597.25	r) 07777	aa) ‘c’

Bài tập luyện tập

Bài 3: Giả sử a, b, c là các biến kiểu int với a = 8, b = 3 và c = 5. Xác định giá trị trả về của các biểu thức sau:

a) a%c*2	f) c * (b / a)
b) a * (a % b)	g) 16*b / a *(b – 1)
c) 2 * b + 3 * (a – c)	h) a/b/c
d) a * (b + (c – 4 * 3))	i) (a * b) % c
e) a + c / a	j) 5 % b % c

Bài tập luyện tập

Bài 4: Cho chương trình C với các biến như sau: int i = 8, j = 5; float x = 0.005, y = –0.01; char c = 'c', d = 'd';

(Trong bảng ASCII giá trị nguyên của các kí tự c và d lần lượt là 99 và 100)
Hãy xác định giá trị trả về của các biểu thức sau:

a) (3 * i – 2 * j) % (4 * d – c)
b) c < d
c) 2 * ((i / 4) + (6 * (j – 3)) % (i + j – 4))
d) (i – 7 * j) % (c + 3 * d) / (x – y)
e) x < y
f) – (i + j) * –1
g) j != 6
h) ++i
i) c = = 99
j) i++
k) i++ + 5
l) 5 * (i + j + 1) > 'd'
m) ++i + 5
n) (3 * x + y) == 0
o) 2 * x + (y == 0)
p) !(i < j)
q) j– – +i
r) (i > 0) && (j < 6)
s) (x > y) && (i > 0) || (j < 5)

Bài tập luyện tập

Bài 5: Cho chương trình có các khai báo biến và khởi tạo như sau:

```
int i = 8, j = 5, k;
float x = 0.005, y = -0.01, z;
char a, b, c = 'c', d = 'd';
```

Xác định giá trị các biểu thức gán sau:

a) i %= j
b) a = b = d
c) i += (j – 3)
d) y– =x
e) k = (j = = 5) ? i : j
f) x *= 2
g) k = (j > 5) ? i : j
h) k = c
i) i /= j
j) i += j *= i /= 2
k) i += i+= i++
l) a = (c < d) ? c : d
m) z = (x >= 0) ? x : 0
n) i - = (j > 0) ? j : 0
o) i = (i*9*(3+(8*j/3)))

Bài tập luyện tập

Bài 6: Tính giá trị của các biểu thức sau:

a) 5 && (7>8)
b) !3 || (2<7)
c) !(4==5)&&(3>7)
d) 3+2%5>10 && (7/3 >10 || 8%3==2)
e) 4+2*3^2-4 > 10 && (1+2^2-8/4 > 6 && (2<6 || 10>11))
f) 5+7 > 8 ? 14%6 : 3*2
g) 20&15
h) 20|15
i) 20^15
k) ~20
m) 20>>2
n) 20<<2