Objectives

At the end of this session, you will be able to:

- *Explain Assignment Operators*
- *Understand Arithmetic Expressions*
- *Explain Relational and Logical Operators*
- *Understand Bitwise Logical Operators and Expressions*
- *Explain Casts*
- *Understand the Precedence of Operators*

C defines four classes of operators: arithmetic, relational, logical, and bitwise. *C định nghĩa bốn lớp toán tử: toán học, so sánh, logic và theo bit.*

Operators operate on constants or variables, which are called operands. C*ác toán tử hoạt động trên các hằng số hoặc biến, được gọi là toán hạng*

```
c = a + b;
```

Here a, b, c are the operands and '=' and '+' are the operators *(Ở đây a, b, c là các toán hạng và '=' và '+' là các toán tử.)*

4.1 Expressions

Biểu thức

An expression can be any combination of operators and operands. Một biểu thức có thể là bất kỳ sự kết hợp nào của các toán tử và toán hạng.

Operators perform operations like addition, subtraction, comparison, etc. Các toán tử thực hiện các phép toán như cộng, trừ, so sánh, v.v. Operands are the variables or values on which the operations are performed. Toán hạng là các biến hoặc giá trị mà trên đó các phép toán được thực hiện.

The whole thing together is an expression.Toàn bộ kết hợp này được gọi là một biểu thức.

simplify using the usual rules (các quy tắc thông thường): parentheses (or brackets) first (trước tiên là ngoặc đơn), then exponents (số mũ), multiplication (phép nhân) and division (phép chia), then addition (phép cộng) and subtraction (phép trừ).

The Assignment Operator

_ toán tử gán

The general form of the assignment operator is: Công thức tổng quát của toán tử gán là:

```
variable_name = expression;
```

Many variables can be assigned the same value in a single statement (Nhiều biến có thể được gán cùng một giá trị trong một câu lệnh duy nhất)

Arithmetic Expressions _biểu thức số học

C using the arithmetic operators with numeric and character operands (sử dụng các toán tử số học với các toán hạng số và ký tự. Các biểu thức như vậy được gọi là Biểu thức số học).

4.2 Relational Operators and Expressions

Toán tử quan hệ và biểu thức

Relational operators are used to test the relationship between two variables, or between a variable and a constant (Toán tử quan hệ được sử dụng để kiểm tra mối quan hệ giữa hai biến hoặc giữa một biến và một hằng số.)

In C, 0 for false and 1 for true.

Operator	Relational Operators Action
>	Greater than
> =	Greater than or equal
<	Less than
< =	Less than or equal
= =	Equal
!=	Not equal

Table 4.1: Relational operators and their action

4.3 Logical Operators and expressions

Toán tử logic và biểu thức

Logical operators are symbols that are used to combine or negate expressions containing relational operators (Toán tử logic là các ký hiệu được sử dụng để kết hợp hoặc phủ định các biểu thức chứa toán tử quan hệ).

Operator	Logical Operators Action
&&	A N D
||	O R
!	NOT

Table 4.2: Logical operators and their action

This AND operator is represented by &&, and the condition will be written as (AND này toán tử được biểu diễn bằng && và điều kiện sẽ được viết như sau):

```
(a < 10) && (b == 7);
```

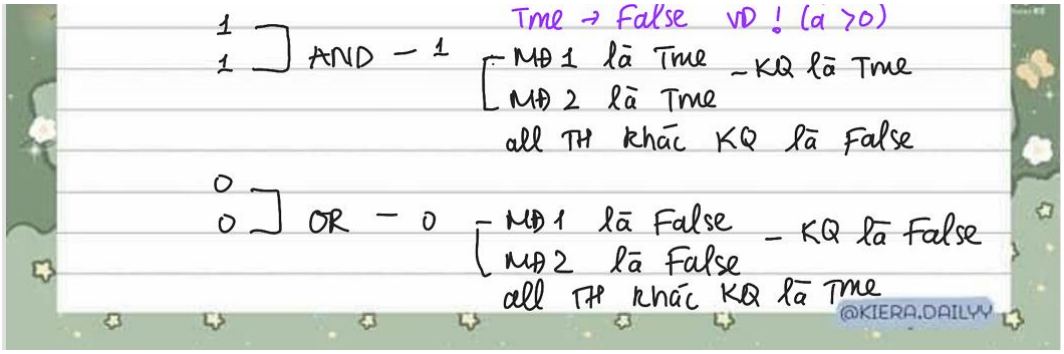
Similarly the OR is operator used to check whether one of the conditions is true (toán tử OR được sử dụng để kiểm tra xem một trong các điều kiện có đúng không.). It is represented by two consecutive pipe signs (||) (Nó được biểu diễn bằng hai dấu ống liên tiếp (||)).

if either of the statements are true then the condition will be coded as (nếu một trong hai câu lệnh là đúng thì điều kiện sẽ được mã hóa như sau):

```
(a < 10) || (b == 7);
```

The third logical operator NOT is represented by a single exclamation mark (!) (Toán tử logic thứ ba NOT được biểu diễn bằng một dấu chấm than (!).). This operator reverses the true value of the expression (Toán tử này đảo ngược giá trị thực của biểu thức).

Toán tử logic bitwise và biểu



4.4 Bitwise Logical Operators and Expressions

Bitwise operators treat the operands as bits rather than numerical values. Các toán tử bitwise xem xét các toán hạng như là các bit thay vì các giá trị số. The numerical values could be of any base: decimal, octal, or hexadecimal. Các giá trị số có thể thuộc bất kỳ hệ số nào: thập phân, bát phân hoặc thập lục phân. The Bitwise operator will convert the operand to its binary representation and accordingly return 1 or a 0. Toán tử bitwise sẽ chuyển đổi toán hạng thành biểu diễn nhị phân của nó và trả về 1 hoặc 0 tương ứng.

Operation	Description
AND (N1 & N2)	Mỗi vị trí trả về 1 nếu các bit đều là 1
OR (N1 | N2)	Mỗi vị trí trả về 1 nếu 1 trong 2 bit là 1
NOT (~ N1)	Đảo ngược các bit của cái toán hạng
XOR (N1 ^ N2)	Mỗi vị trí trả về 1 nếu 1 trong 2 bit là 1 nhưng Ko phải cả 2
(a << n)	Dịch các bit sang trái, thêm bit 0 bên phải Thường sd nhân 1 số vs lũy thừa 2
(a >> n)	Dịch phải

4.5 Mixed Mode Expressions & Type Conversions

_ Biểu thức chế độ hỗn hợp & Chuyển đổi kiểu

All operands are converted to the data type of the largest operand. Tất cả các toán hạng được chuyển đổi về kiểu dữ liệu của toán hạng lớn nhất. This is called type promotion. Điều này được gọi là khuyến khích kiểu dữ liệu.

The order of the various data types is: Thứ tự của các kiểu dữ liệu khác nhau là

```
char < int < long < loat < double
```

The automatic type conversions for evaluating an expression are tabulated below (Các chuyển đổi kiểu tự động để đánh giá một biểu thức được liệt kê dưới đây)

a. char and short are converted to int, and float is converted to double (char và short được chuyển đổi thành int, và float được chuyển đổi thành double)
b. If either operand is double, the other is converted to double, and the result is double (Nếu một trong hai toán hạng là double, toán hạng kia sẽ được chuyển đổi thành double và kết quả là double)
c. If either operand is long, the other is converted to long, and the result is double (Nếu một trong hai toán hạng là dài, toán hạng kia sẽ được chuyển thành dài và kết quả là double)
d. If either operand is unsigned, the other is also converted to unsigned, and the result is also unsigned (Nếu một trong hai toán hạng không có dấu, toán hạng kia cũng được chuyển đổi thành không có dấu và kết quả cũng không có dấu.)
e. Otherwise all that are left are the operands of type int, and the result is in (Nếu không thì tất cả những gì còn lại là các toán hạng kiểu int và kết quả là int)

when one operand is long and the other is unsigned (một toán hạng là long và toán hạng kia là unsigned,), both operands are converted to unsigned long (cả hai toán hạng đều được chuyển đổi thành unsigned dài)

4.5.1 Casts

ép kiểu

The general syntax of cast is:

```
(type) cast
```

where type is a valid C data type

```
int i = 1, j = 3;
x = i / j; /* x 0.0 */
x = (float) i/(float) j; /* x 0.33 */
```

Thứ tự ưu tiên của các toán tử

4.6 Precedence of Operators

Operator Class	Operators	Associativity
Unary Binary	- ++ – – ^	Right to left Left to Right
Binary	* / %	Left to Right
Binary	+ -	Left to Right
Binary	=	Right to Left

Table 4.4: Order of precedence of the arithmetic operators

If there are several sets of parentheses in an expression then evaluation takes place from left to right (Nếu có nhiều bộ dấu ngoặc đơn trong một biểu thức thì việc đánh giá sẽ diễn ra từ trái sang phải.)

The assignment operator (=) associates from right to left. Toán tử gán (=) liên kết từ phải sang trái. Hence, the expression on the right is evaluated first and its value is assigned to the variables on the left. Do đó, biểu thức bên phải được đánh giá trước tiên và giá trị của nó được gán cho các biến bên trái.

the unary minus is evaluated first as it has highest precedence (phép trừ đơn được đánh giá đầu tiên vì nó có mức độ ưu tiên cao nhất). The evaluation of * and % takes place from left to right (Đánh giá * và % diễn ra từ trái sang phải). This is followed by evaluation of the binary – operator (Tiếp theo là việc đánh giá toán tử nhị phân).

Precedence between Comparison Operators: There is no such precedence among comparison operators (Không có loại ưu tiên nào như vậy giữa các toán tử so sánh). They are therefore always evaluated left to right (Do đó, chúng luôn được đánh giá từ trái sang phải.)

Precedence for Logical Operators

Precedence	Operator
1	NOT
2	A N D
3	OR

Table 4.5: Order of precedence for logical operators

Precedence among the Different Types of Operators

Precedence	Type of Operator
1	Arithmetic
2	Comparison
3	Logical

Table 4.6: Order of precedence among different types of operators

The Parentheses (or brackets) (Dấu ngoặc đơn (hoặc dấu ngoặc vuông))

- When there are parentheses within parentheses, the innermost parentheses are to be evaluated first (Khi có dấu ngoặc đơn bên trong dấu ngoặc đơn, dấu ngoặc đơn trong cùng sẽ được đánh giá trước).
- When an equation involves multiple sets of parentheses, they are evaluated left to right (Khi một phương trình có nhiều cặp dấu ngoặc đơn, chúng sẽ được tính từ trái sang phải.)

Summary

- C defines four classes of operators: arithmetic, relational, logical, and bitwise.
- All operators in C follow a precedence order.
- Relational operators are used to test the relationship between two variables, or between a variable and a constant.
- Logical operators are symbols that are used to combine or negate expressions containing relational operators.
- Bitwise operators treat the operands as bits rather than numerical value.
- Assignment (=) is considered an operator with right to left associativity.
- Precedence establishes the hierarchy of one set of operators over another when an expression has to be evaluated.

Ex1 False OR True AND NOT False AND True
Trong đó 1_ NOT , 2_ AND , 3_ OR

NOT : đảo ngược giá trị => NOT False = True

Viết lại False OR True AND True AND True
2_ AND : từ phải qua trái:
True AND True → True

T
T] AND – T

Viết lại False OR True AND True
True AND True → True

Viết lại False OR True

F
F] OR – F

↓

TRUE

Ex2: 2* 3 + 4 / 2 > 3 AND 3 < 5 OR 10 < 9
Trong đó 1_ arithmetic , 2_ relational , 3_ logic
2 * 3 + 4 / 2
nhân chia trước => 6 + 2
cộng trừ sau => 8
Viết lại 8 > 3 AND 3 < 5 OR 10 < 9
2_ relational 8 > 3 : True , 3 < 5 : True , 10 < 9 : False
Viết lại T AND T OR F
1_ NOT, 2_ AND, 3_ OR
T AND T → T
Viết lại T OR F

T
T] AND – T

F
F] OR – F

↓

T

Ex3: 5+9*3^2-4 > 10 AND (2+2^4-8|4 > 6 OR (2<6 AND 10>11))

Ngoặc đơn (ngoặc vuông) → lũy thừa → nhân, chia → cộng trừ

2<6 → TRue
10>11 → False
=) True AND False => False

Xét () thứ 2 (2+2^4-8|4 >6 OR False)

$2^4 = 16$ =) 2 + 16 - 8|4

=) 2 + 16 - 2 => 16 > 6 => True

=) True OR False

Xét 5+9*3^2-4 =) $3^2 = 9$ => 5+9*9-4

=) 5+81-4 =) 82

Viết lại 82 > 10 AND True

True AND True => True

Toán tử 1 ngôi: a + b = c

- abc : toán hạng
- =, + : toán tử

VD: a = 10 , b = a

Case 1: a ++ => Cộng sau thì thêm sau
b = a = 10
a = 11 (a += 1)

Case 2: ++a => Cộng trước thì thêm trước
a = 11 (a += 1)
b = a = 11

Ex02: ép kiểu dữ liệu

gán int n_1 | printf ("....", n_1, n_2, n_1 + (int) n_2);
float n_2

Ex03: Cấu trúc hàm printf (), scanf ()

- printf () - hàm sd hiển thị dữ liệu đầu ra

printf (" control string", argument list);

control string - chuỗi điều kiện - nằm trong ngoặc kép
- bao gồm: text character - kí tự văn bản
format commands - lệnh định dạng
non character - kí tự ko in
(tab, blanks, new lines)

Ex04: special character - kí tự đặc biệt

	printf
\\	\
\	" to printf "
%%	%

Ex05: modifier – bộ điều chỉnh
- modifier – mục dữ liệu đc căn trái
Field Width Modifier – bộ sửa đổi độ rộng trường
là 1 số nguyên, xđ độ rộng trường tối thiểu
VD %10f – lệnh định dạng kiểu float có chiều rộng 10
Precision Modifier – bộ sửa đổi chính xác
đc viết .m trong đó m là số nguyên
VD %10.3f – lệnh định dạng kiểu float có chiều rộng 10
và 3 vị trí sau dấu thập phân
'0' Modifier – đệm mặc định thực hiện = khoảng trắng
'l' Modifier – hiển thị các số nguyên dưới dạng long int
hoặc đối số có độ chính xác kép. Mặc định dạng %ld
'h' Modifier – hiển thị các số nguyên ngắn. Mặc định dạng %hd
'*' Modifier – sd nếu ng dùng Ko muốn chỉ định trước độ rộng trường
nhưng muốn CT chỉ định

Ex08: getchar() | CT nhập xuất ký tự
putchar() | đơn giản
getchar() – đọc 1 ký tự từ bàn phím
Syntax (cú pháp) int getchar(void);
Cách hoạt động – trả về mã ASCII của ký tự
– giá trị là int
VD: ch = 'A'; ch = getchar(); | output: 97
putchar(ch);
putchar() – in 1 ký tự ra màn hình
Syntax int putchar(int ch);
Cách hoạt động – nhận mã ASCII
– trả về chính ký tự đó
VD: putchar(70); | Output: F