



C PROGRAMING

by Ketan Kore @ Sunbeam Infotech



Operators Precedence and Associativity

OPERATOR	TYPE	ASSOCIIVITY
() [] . ->		left-to-right
++ -- +- ! ~ (type) * & sizeof	Unary Operator	right-to-left
* / %	Arithmetic Operator	left-to-right
+ -	Arithmetic Operator	left-to-right
<< >>	Shift Operator	left-to-right
< <= > >=	Relational Operator	left-to-right
== !=	Relational Operator	left-to-right
&	Bitwise AND Operator	left-to-right
^	Bitwise EX-OR Operator	left-to-right
	Bitwise OR Operator	left-to-right
&&	Logical AND Operator	left-to-right
	Logical OR Operator	left-to-right
? :	Ternary Conditional Operator	right-to-left
= += -= *= /= %= &= ^= = <=> >>=	Assignment Operator	right-to-left
,	Comma	left-to-right



Twisters

- If precedence of two operators in an expression is same, their associativity is considered to decide their binding with operands.
- Data type conversions and ranges should be considered while doing arithmetic operations.
- `sizeof()` is compile time operator. Expressions within `sizeof` are not executed at runtime.
- Relational and logical operators always result in 0 or 1.
- In logical AND, if first condition is false, second condition is not evaluated. Result is false.
- In logical OR, if first condition is true, second condition is not evaluated. Result is true.
- Increment/Decrement operators in arithmetic expressions are compiler dependent.



Control Statements

- Decision or Selection
 - if-else
 - switch-case
- Iteration (loop)
 - for
 - while
 - do-while
- Jump
 - break
 - continue
 - goto
 - return



Selection Statement

- If statement
- If-else statement
- Switch statement

- If statement

- General form of if statement is

```
    If(expression) // if header  
        statement // if body
```

- If the if controlling expression is evaluated to true the statement constituting if body is executed
 - There should be no semicolon at the end of if header

- If-else

- Most of the problems require one set of action to be performed if particular condition is true and another set of action to be performed if condition is false
 - C language provides an if-else statement
 - General form is

```
    If(expression)  
        statement 1;  
    else  
        statement 2;
```



- Nested if Statement

- If the body of if statement contains another if statement then we say that if are nested and the statement is known as nested if statement

- If(expression)
 {
 statement

 if statement

 }

- Nested if- else statement

- In nested if-else statement the if body or else body of an if-else statement contains another if statement or if else statement



if-else statement

```
if (condition) {  
    statement 1;  
    statement 2;  
}
```

```
if (condition) {  
    statement 1;  
    statement 2;  
}  
else {  
    statement 3;  
    statement 4;  
}
```

```
if (condition)  
    statement 1;
```

```
if (condition)  
    statement 1;  
else  
    statement 2;
```

- Condition is any expression – using relational, logical or other operators.
 - 0 – false condition
 - 1 – true condition



Ternary/conditional operator

```
if (condition) {  
    // execute if condition is true  
}  
else {  
    // execute if condition is false  
}
```

- if-else can be nested within each other.

condition ? expression1 : expression2

- If condition is true, expression1 is executed; otherwise expression2 is executed.
- Ternary operators can also be nested.
- **expression1 & expression2 must be expressions (not statement).**
 - expression – evaluate to some value.
 - statement – C statement ends with ;



Switch statement

- .A switch statement is used to control Complex branching operations , When there are many Conditions it becomes too difficult to use if and if-else
- In such case switch provides easy and organized way to select among multiple operations
- General form of switch statement is
 - Switch (expression)
statement
- Switch selection expression must be of integral type
- Case labeled constituting the body of switch should be unique i.e. no two case labels should evaluate to same value



switch-case

```
switch (expression) {  
    case const-expr1:  
        statement(s);  
        break;  
    case const-expr2:  
        statement(s);  
        break;  
    ...  
    default:  
        statement(s);  
        break;  
}
```

- Switch-case is used to select one of the several paths to execute depending on value of **int expression**.
- case constants **cannot be duplicated**.
- break statement skips remaining statements and continues execution at the end of switch closing brace.
- If break is missing, statements under sub-subsequent case continue to execute.
- default case is optional and it is executed only if int expression is not matching with any of the case constant.
- **Sequence of cases and default case doesn't matter.**





Thank you!

Ketan Kore<Ketan.Kore@sunbeaminfo.com>

