Operators and Decision-Making Constructs

Objectives

In this lesson, you will learn to:

- Use the following operators:
  - Arithmetic
  - Assignment
  - Unary
  - Comparison
  - Logical
- Use decision-making constructs
  - if…else constructs
  - switch…case constructs

Objectives (Contd.)

- Use loop constructs
  - while loop
  - do…while loop
  - for loop
- Scope of variables

Operators and Decision-Making Constructs

Types of Operators

- Arithmetic operators
- Assignment operators
- Unary operators
- Comparison operators
- Logical operators
- Conditional operators

Operators and Decision-Making Constructs

Arithmetic Operators

- Are used to perform arithmetic operations
- Are used in an order according to the precedence rules

Type	Operators	Associativity
Value construction	( )	Innermost to outermost
Multiplicative	* / %	Left to right
Additive	+ -	Left to right

Operators and Decision-Making Constructs

Arithmetic Assignment Operators

- Are used to assign the value of the right operand to the left
- Are as follows: =, +=, -=, /=, *=, %=

Operators and Decision-Making Constructs

Unary Operators

- Operate on one operand
- Are of two types:
  - Increment operator (++)
  - Decrement operator (--)

Operators and Decision-Making Constructs

Comparison Operators

- Are also called relational operators
- Are used to compare two values
- Evaluate to `true` or `false`

Operators and Decision-Making Constructs

Comparison Operators (Contd.)

Operator	Description	Example	Explanation
==	Evaluates whether or not the operands are equal.	x == y	Returns **true** if the values are equal and **false** otherwise
!=	Evaluates whether or not the operands are not equal	x != y	Returns **true** if the values are not equal and **false** otherwise
>	Evaluates whether or not the left operand is greater than the right operand	x > y	Returns **true** if x is greater than y and **false** otherwise
<	Evaluates whether or not the left operand is less than the right operand	x < y	Returns **true** if x is less than y and **false** otherwise
>=	Evaluates whether or not the left operand is greater than or equal to the right operand	x >= y	Returns **true** if x is greater than or equal to y and **false** otherwise
<=	Evaluates whether or not the left operand is less than or equal to the right operand	x <= y	Returns **true** if x is less than or equal to y and **false** otherwise

Operators and Decision-Making Constructs

Logical Operators

- Are used to combine two or more expressions
- Are of three types:
  - AND (&&)
  - OR (||)
  - NOT (!)

Operators and Decision-Making Constructs

Conditional Constructs

- Control the flow of a program
- Allow selective execution of statements
- Use comparison operators for evaluating conditions
- Are of two types:
  - The `if…else` construct
  - The `switch…case` construct

Operators and Decision-Making Constructs

The if…else Construct

In an `if…else` block of statements:

- Condition is evaluated first
- If condition is true, statements in the immediate block are executed
- If condition is false, statements in the `else` block are executed

The if…else Construct (Contd.)

Syntax:

```
if (expression)
{
statements;
}
else
{
statements;
}
```

Operators and Decision-Making Constructs

Practice :

Write a construct that assigns grades to students based on their marks. Students who have scored marks between 75 and 100 are to be given grade ‘A’, those who have scored between 50 and 75 are to be given grade ‘B’, and the rest of them should be given grade ‘C’.

The switch…case Construct

- Is used when there are multiple values for a variable
- Evaluates condition-variable of the `switch` statement and compares with each `case` constant
- Requires a `break` statement to exit from its body
- Uses `default` keyword for associating any other statements other than mentioned in the `case` constant

Operators and Decision-Making Constructs

Loop Constructs

- Cause a section of a program to be repeated a certain number of times
- Are of three types:
  - `while` loop
  - `do…while` loop
  - `for` loop

Operators and Decision-Making Constructs

The while Loop

- Continues until the evaluating condition becomes false

Syntax:

```
while (expression)
{
statements;
}
```

- Requires the `continue` statement to return the control to the beginning of while loop skipping any other statements after the `continue` keyword

Operators and Decision-Making Constructs

Practice:

Write a function to display the sum of all numbers between 1 and 100.

The do...while Loop

- Continues until the evaluating condition becomes false
- Executes the body of the loop at least once

Syntax:

```
do
{
statements;
} while(boolean_expr);
```

Operators and Decision-Making Constructs

The for Loop

- Provides a compact way of specifying statements that control the repetition of the steps within the loop
- Contains three expressions:
  - The initialization expression
  - The test expression
  - The increment/decrement expression

The for Loop (Contd.)

Syntax:

```
for( initialization_expr; test_expr; change_expr)
{
        statements;
}
```

Operators and Decision-Making Constructs

The break and continue statement

- The `break` statement is used to exit a loop before the loop condition is re-evaluated after iteration
- The `continue` statement is used to skip all the subsequent instructions and take the control back to the loop

Operators and Decision-Making Constructs

Problem Statement

Write a program that will reverse an accepted string and copy it into another string.

Problem Statement

Write a program that displays the amount outstanding for all customers. The amount outstanding should be displayed in an ascending order.

The amount outstanding is represented as an array of `float` values:

```
float amounts[10] =
{200.5,323,0,100.7,314,523,256,10.90,
   553.90,0};
```

Operators and Decision-Making Constructs

Problem

Write a program to accept the salaries of 10 employees from the user and displays them in descending order for all the employees. If the user enters zero, the program should display the message “The amount should be greater than zero” and accept the value again.

The Scope of a Variable

- Falls under three heads:
  - File scope
  - Local scope
  - Class scope

Operators and Decision-Making Constructs

File Scope

- Is considered to be the outermost scope
- Variables are accessible throughout the program file
- Are called *global variables*

Local Scope

- Is defined as being limited to the braces of a function or a control structure like `for`, `while`, and `if`
- Are called *local variables*

Operators and Decision-Making Constructs

Class Scope

- Variables are accessible only within the class
- Variables have the flexibility of being accessed outside the class by declaring them as `public`

Operators and Decision-Making Constructs

Summary

In this lesson, you learned that:

- Operators are used to compute and compare values and test multiple conditions
- You use arithmetic operators to perform arithmetic operations, such as addition, subtraction, multiplication, and division
- You can also use arithmetic assignment operators to perform arithmetic operations
- The unary operators, such as the increment and decrement operators operate on one operand
- Comparison operators, which are also called relational operators, are used to compare two values

Summary (Contd.)

- Logical operators are used to combine two or more expressions
- Conditional constructs are used to allow the selective execution of statements. The conditional constructs in C++ are:
  - `if...else`
  - `switch...case`
- Looping constructs are used when you want a section of a program to be repeated a certain number of times. C++ offers the following looping constructs:
  - `while`
  - `do...while`
  - `for`

Summary (Contd.)

- The `break` and `continue` statements are used to control the program flow within a loop
- The scope of a variable specifies its accessibility
- C++ features three types of scopes: file, class, and local scope