# Code to success review session

DR. HADJI J. TEJUCO

# TOPICS

- Logic and Design Formulation
- Basic Input and Output
- Conditional Control Structure
- Repetition Control Structure
- Function and Recursion

# Logic and Design Formulation

ALL PROGRAMMING INVOLVES CREATING SOMETHING THAT SOLVES A PROBLEM.

IDENTIFY THE PROBLEM

DESIGN A SOLUTION

WRITE THE PROGRAM

CHECK THE SOLUTION

# Basic Input and Output

- **C++ Output**
- In C++, cout sends formatted output to standard output devices, such as the screen. We use the cout object along with the << operator for displaying output.

- **C++ Input**
- In C++, cin takes formatted input from standard input devices such as the keyboard. We use the cin object along with the >> operator for taking input.

- Cascading of Insertion ( << ) operator and Cascading of Extraction ( >> ) operator

# C++ Fundamental Data Types

The table below shows the fundamental data types, their meaning, and their sizes (in bytes):

| Data Type | Meaning | Size (in Bytes) |
|---|---|---|
| `int` | Integer | 2 or 4 |
| `float` | Floating-point | 4 |
| `double` | Double Floating-point | 8 |
| `char` | Character | 1 |
| `wchar_t` | Wide Character | 2 |
| `bool` | Boolean | 1 |
| `void` | Empty | 0 |

# C++ Operators

- Arithmetic Operators
- Assignment Operators
- Relational Operators
- Logical Operators

| Operator | Operation |
| --- | --- |
| + | Addition |
| - | Subtraction |
| * | Multiplication |
| / | Division |
| % | Modulo Operation (Remainder after division) |

# C++ Arithmetic Operators

# Increment and Decrement Operators

- C++ also provides increment and decrement operators: ++ and -- respectively. ++ increases the value of the operand by **1**, while -- decreases it by **1**.

| Operator | Example | Equivalent to |
|---|---|---|
| `=` | `a = b;` | `a = b;` |
| `+=` | `a += b;` | `a = a + b;` |
| `-=` | `a -= b;` | `a = a - b;` |
| `*=` | `a *= b;` | `a = a * b;` |
| `/=` | `a /= b;` | `a = a / b;` |
| `%=` | `a %= b;` | `a = a % b;` |

# C++ Assignment Operators

| Operator | Meaning | Example |
| --- | --- | --- |
| `==` | Is Equal To | `3 == 5` gives us **false** |
| `!=` | Not Equal To | `3 != 5` gives us **true** |
| `>` | Greater Than | `3 > 5` gives us **false** |
| `<` | Less Than | `3 < 5` gives us **true** |
| `>=` | Greater Than or Equal To | `3 >= 5` give us **false** |
| `<=` | Less Than or Equal To | `3 <= 5` gives us **true** |

# C++ Relational Operators

| Operator | Example | Meaning |
| --- | --- | --- |
| `&&` | expression1 **&&** expression2 | Logical AND.<br>True only if all the operands are true. |
| `\|\|` | expression1 **\|\|** expression2 | Logical OR.<br>True if at least one of the operands is true. |
| `!` | **!**expression | Logical NOT.<br>True only if the operand is false. |

# C++ Logical Operators

# Conditional Control Structure

- IF
- IF ELSE
- IF ELSE IF ELSE
- NESTED IF ELSE
- SWITCH STATEMENT

# C++ if Statement

The syntax of the `if` statement is:

```
if (condition) {
   // body of if statement
}
```

The `if` statement evaluates the `condition` inside the parentheses `( )`.

- If the `condition` evaluates to `true`, the code inside the body of `if` is executed.
- If the `condition` evaluates to `false`, the code inside the body of `if` is skipped.

> **Note:** The code inside `{ }` is the body of the `if` statement.

The `if` statement can have an optional `else` clause. Its syntax is:

```
if (condition) {
    // block of code if condition is true
}
else {
    // block of code if condition is false
}
```

The `if..else` statement evaluates the `condition` inside the parenthesis.

# C++ if...else

The syntax of the `if...else if...else` statement is:

```
if (condition1) {
    // code block 1
}
else if (condition2){
    // code block 2
}
else {
    // code block 3
}
```

# C++ if...else...else if statement

```
// outer if statement
if (condition1) {
    // statements

    // inner if statement
    if (condition2) {
        // statements
    }
}
```

# C++ Nested if...else

The `switch` statement allows us to execute a block of code among many alternatives.

The syntax of the `switch` statement in C++ is:

```
switch (expression)  {
    case constant1:
        // code to be executed if
        // expression is equal to constant1;
        break;

    case constant2:
        // code to be executed if
        // expression is equal to constant2;
        break;
        .
        .
        .
    default:
        // code to be executed if
        // expression doesn't match any constant
}
```

# C++ switch..case Statement

# Repetition Control Structure

- FOR
- WHILE
- DO WHILE

## C++ for loop

The syntax of for-loop is:

```
for (initialization; condition; update) {
    // body of-loop
}
```

Here,

- `initialization` - initializes variables and is executed only once
- `condition` - if `true`, the body of `for` loop is executed
  if `false`, the for loop is terminated
- `update` - updates the value of initialized variables and again checks the condition

## C++ while Loop

The syntax of the `while` loop is:

```
while (condition) {
    // body of the loop
}
```

Here,

- A `while` loop evaluates the `condition`
- If the `condition` evaluates to `true`, the code inside the `while` loop is executed.
- The `condition` is evaluated again.
- This process continues until the `condition` is `false`.
- When the `condition` evaluates to `false`, the loop terminates.

## C++ do...while Loop

The `do...while` loop is a variant of the `while` loop with one important difference: the body of `do...while` loop is executed once before the `condition` is checked.

Its syntax is:

```
do {
   // body of loop;
}
while (condition);
```

Here,

- The body of the loop is executed at first. Then the `condition` is evaluated.
- If the `condition` evaluates to `true`, the body of the loop inside the `do` statement is executed again.
- The `condition` is evaluated once again.
- If the `condition` evaluates to `true`, the body of the loop inside the `do` statement is executed again.
- This process continues until the `condition` evaluates to `false`. Then the loop stops.

# Function and Recursion

There are two types of function:

**Standard Library Functions:** Predefined in C++

**User-defined Function:** Created by users

Function with no argument and no return value

Function with no argument but return value

Function with argument but no return value

Function with argument and return value

Program to explain Default Arguments

Program to explain Function Overloading

# C++ Function Declaration

The syntax to declare a function is:

```
returnType functionName (parameter1, parame
    // function body
}
```