Structured Programming
Using C++

Lecture 3: Flow of Control (I)

Dr. Amal Khalifa

Quiz

What will be the value of each of the following variables after its initialization??

double d = 2 * int(3.14);
long k = 3.14 - 3;
char c = 'a' + 2;
char c = 'p' + 'A' - 'a';
Lecture Contents:

- Boolean Expressions
  - Building, Evaluating & Precedence Rules
- Branching Mechanisms
  - if-else
  - switch
  - Nesting if-else

### Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>≥</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>≤</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>≠</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
Boolean Expressions:

Display 2.1 Comparison Operators

<table>
<thead>
<tr>
<th>MATH SYMBOL</th>
<th>ENGLISH</th>
<th>C++ NOTATION</th>
<th>C++ SAMPLE</th>
<th>MATH EQUIVALENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
<td>==</td>
<td>x + 7 == 2*y</td>
<td>x + 7 * 2y</td>
</tr>
<tr>
<td>≠</td>
<td>Not equal to</td>
<td>!=</td>
<td>ans != 'n'</td>
<td>ans ≠ 'n'</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>&lt;</td>
<td>count &lt; m + 3</td>
<td>count &lt; m * 3</td>
</tr>
<tr>
<td>≤</td>
<td>Less than or</td>
<td>&lt;=</td>
<td>time &lt;= limit</td>
<td>time ≤ limit</td>
</tr>
<tr>
<td></td>
<td>equal to</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>&gt;</td>
<td>time &gt; limit</td>
<td>time &gt; limit</td>
</tr>
<tr>
<td>≥</td>
<td>Greater than</td>
<td>&gt;=</td>
<td>age &gt;= 21</td>
<td>age ≥ 21</td>
</tr>
</tbody>
</table>

Evaluating Boolean Expressions:

Display 2.2 Truth Tables

**AND**

<table>
<thead>
<tr>
<th>Expr₁</th>
<th>Expr₂</th>
<th>Expr₁ &amp; Expr₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

**NOT**

<table>
<thead>
<tr>
<th>Exp</th>
<th>!Exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

**OR**

| Expr₁ | Expr₂ | Expr₁ || Expr₂ |
|-------|-------|---------|
| true  | true  | true    |
| true  | false | true    |
| false | true  | true    |
| false | false | false   |
### Precedence of Operators (1 of 4)

Display 3.3 Precedence of Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>::</td>
<td>Scope resolution operator</td>
</tr>
<tr>
<td>.</td>
<td>Dot operator</td>
</tr>
<tr>
<td>-&gt;</td>
<td>Member selection</td>
</tr>
<tr>
<td>[]</td>
<td>Array indexing</td>
</tr>
<tr>
<td>()</td>
<td>Function call</td>
</tr>
<tr>
<td>++</td>
<td>Prefix increment operator (placed before the variable)</td>
</tr>
<tr>
<td>--</td>
<td>Prefix decrement operator (placed before the variable)</td>
</tr>
<tr>
<td>!</td>
<td>Not</td>
</tr>
<tr>
<td>-</td>
<td>Unary minus</td>
</tr>
<tr>
<td>+</td>
<td>Unary plus</td>
</tr>
<tr>
<td>*</td>
<td>Dereference</td>
</tr>
<tr>
<td>&amp;</td>
<td>Address of</td>
</tr>
<tr>
<td>new</td>
<td>Create (allocate memory)</td>
</tr>
<tr>
<td>delete</td>
<td>Destroy (deallocate)</td>
</tr>
<tr>
<td>delete[]</td>
<td>Destroy array (deallocate)</td>
</tr>
<tr>
<td>sizeof</td>
<td>Size of object</td>
</tr>
<tr>
<td>()</td>
<td>Type cast</td>
</tr>
</tbody>
</table>

### Precedence of Operators (2 of 4)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Multiply</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
</tr>
<tr>
<td>%</td>
<td>Remainder (modulo)</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td>&lt;&lt;</td>
<td>Insertion operator (console output)</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>Extraction operator (console input)</td>
</tr>
</tbody>
</table>
Precedence of Operators (3 of 4)

Display 2.3  Precedence of Operators

All operators in part 2 are of lower precedence than those in part 1.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>==</td>
<td>Equal</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>And</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Precedence of Operators (4 of 4)

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Assignment</td>
</tr>
<tr>
<td>+=</td>
<td>Add and assign</td>
</tr>
<tr>
<td>-=</td>
<td>Subtract and assign</td>
</tr>
<tr>
<td>*=</td>
<td>Multiply and assign</td>
</tr>
<tr>
<td>/=</td>
<td>Divide and assign</td>
</tr>
<tr>
<td>%=</td>
<td>Modulo and assign</td>
</tr>
<tr>
<td>? :</td>
<td>Conditional operator</td>
</tr>
<tr>
<td>throw</td>
<td>Throw an exception</td>
</tr>
<tr>
<td>,</td>
<td>Comma operator</td>
</tr>
</tbody>
</table>

Lowest precedence (done last)
Precedence Examples

- Arithmetic before logical
  - $x + 1 > 2 \; \text{||} \; x + 1 < -3$ means:
    - $(x + 1) > 2 \; \text{||} \; (x + 1) < -3$

- Short-circuit evaluation
  - $(x \geq 0) \; \&\& \; (y > 1)$
  - Be careful with increment operators!
    - $(x > 1) \; \&\& \; (y++)$

- Integers as boolean values
  - All non-zero values $\rightarrow$ true
  - Zero value $\rightarrow$ false

Compare and Branch

- A program can instruct a computer to compare two items and do something based on a match or mismatch which, in turn, redirect the sequence of programming instructions.
- There are two forms:
  - IF-THEN
  - IF-THEN-ELSE

```
if (Boolean_Expression_1)
    Statement_1;
else
    Statement_2;
```
IF-THEN-ELSE

Entry

false

true

condition

“false” statement

Exit

“true” statement

Branching Mechanisms

- Use { } for multiples statement
  - Also called a "block" stmt
  - Even if just one statement
  - Enhances readability

- else clause is optional
  - If, in the false branch (else), you want "nothing" to happen, leave it out
Example 1: Grades

Write an algorithm to determine a student's final grade and indicate whether it is passing or failing. The final grade is calculated as the average of four marks.

Hint:
- the max grade is 100.
- The student is considered pass if the average $\geq 50\%$

Example 2: Payroll

Write a program that computes the salary of a sales representative. The employee has to work for a minimum number of hours. He will get bonus for each extra hour he works.
Common Pitfalls

- Operator "=" vs. operator "=="
- One means "assignment" (=)
- One means "equality" (==)
  - VERY different in C++!
  - Example:
    ```
    if (x = 12) // Note operator used!
        Do_Something
    else
        Do_Something_Else
    ```

Nested if

- if-else statements contain smaller statements
  - Compound or simple statements (we've seen)
  - Can also contain any statement at all, including another if-else stmt!
Multiway if-else

- Not new, just different indenting
- Avoids "excessive" indenting
- Syntax:

```
Multiway if-else Statement
SYNTAX
if (Boolean_Expression_1)
    Statement_1;
else if (Boolean_Expression_2)
    Statement_2;
   ...
else if (Boolean_Expression_n)
    Statement_n;
else
    Statement_For_All_Other_Possibilities;
```

Example

```
EXAMPLE
if ((temperature < -10) && (day == SUNDAY))
    cout << "Stay home."
else if (temperature < -10)  // and day != SUNDAY
    cout << "Stay home, but call work."
else if (temperature <= 0)   // and temperature >= -10
    cout << "Dress warm."
else // temperature > 0
    cout << "Work hard and play hard."
```

The Boolean expressions are checked in order until the first true Boolean expression is encountered, and then the corresponding statement is executed. If none of the Boolean expressions is true, then the Statement_For_All_Other_Possibilities is executed.
Example 3: Grades revisited

Write a program to determine a student’s final grade according to the following rules:

- A → 90% or more
- B → 80% - 90%
- C → 70% - 80%
- D → 70% - 60%
- F → otherwise

Notice that the final grade is calculated as the average of the four marks, where the max grade is 100.

The switch Statement

```
switch Statement
SYNTAX:
switch (Controlling_Expression)
{
    case Constant_1:
        Statement_Sequence_1
        break;
    case Constant_2:
        Statement_Sequence_2
        break;
        ...
        ...
    case Constant_n:
        Statement_Sequence_n
        break;
    default:
        Default_Statement_Sequence
}
```

You need not place a break statement in each case. If you omit a break, that case continues until a break (or the end of the switch statement) is reached.
The switch Statement in Action

Example 4: Simple calculator

Write a program that accepts two numbers from the user as well as a mathematical operation to perform. The user must choose between the basic four operators: (+, -, *, /).

Additional activity:
Allow the user to write the operation in one line.

For example:

\[ 3 + 4 \]
### switch Pitfalls/Tip

- **Forgetting the break;**
  - No compiler error
  - Execution simply "falls thru" other cases until break;

- **Biggest use: MENUs**
  - Provides clearer "big-picture" view
  - Shows menu structure effectively
  - Each branch is one menu choice

**Example 5:**

```c
switch (response) {
    case "1":
        // Execute menu option 1
        break;
    case "2":
        // Execute menu option 2
        break;
    case "3":
        // Execute menu option 3
        break;
    default:
        cout << "wrong choice !!";
}
```
switch Pitfalls/Tip

- Execution "falls thru" until break
- switch provides a "point of entry"
- Example:
  case "A":
  case "a":
    cout << "Excellent: you got an "A"!\n";
    break;
  case "B":
  case "b":
    cout << "Good: you got a "B"!\n";
    break;
- Note multiple labels provide same "entry"

Conditional Operator

- Also called "ternary operator"
- Allows embedded conditional in expression
- Essentially "shorthand if-else" operator
- Example:
  if (n1 > n2)
    max = n1;
  else
    max = n2;
- Can be written:
  max = (n1 > n2) ? n1 : n2;
  "?" and ":" form this "ternary" operator
That’s all for today !!

Thanks…..