



Programming Fundamentals

Lecture 4

Aamina Batool

Relational Operators

- Relational operators:
 - Allow comparisons
 - Require two operands (binary)
 - Return 1 if expression is `true`, 0 otherwise
- Comparing values of different data types may produce unpredictable results
 - For example, `8 < '5'` should not be done
- Any nonzero value is treated as `true`

TABLE 4-1 Relational Operators in C++

Operator	Description
==	equal to
!=	not equal to
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

Logical (Boolean) Operators

- Logical (Boolean) operators enable you to combine logical expressions
- Three logical (Boolean) operators:
 - ! - not
 - && – and
 - || - or
- Logical operators take logical values as operands and yield logical values as results
- ! is unary; && and || are binary operators
- Putting ! in front of a logical expression reverses its value



TABLE 4-5 Logical (Boolean) Operators in C++

Operator	Description
!	not
&&	and
	or

TABLE 4-6 The ! (Not) Operator

Expression	!(Expression)
<code>true</code> (nonzero)	<code>false</code> (0)
<code>false</code> (0)	<code>true</code> (1)

EXAMPLE 4-2

Expression	Value	Explanation
<code>!('A' > 'B')</code>	<code>true</code>	Because <code>'A' > 'B'</code> is <code>false</code> , <code>!('A' > 'B')</code> is <code>true</code> .
<code>!(6 <= 7)</code>	<code>false</code>	Because <code>6 <= 7</code> is <code>true</code> , <code>!(6 <= 7)</code> is <code>false</code> .

TABLE 4-7 The **&&** (And) Operator

Expression1	Expression2	Expression1 && Expression2
true (nonzero)	true (nonzero)	true (1)
true (nonzero)	false (0)	false (0)
false (0)	true (nonzero)	false (0)
false (0)	false (0)	false (0)

EXAMPLE 4-3

Expression	Value	Explanation
(14 >= 5) && ('A' < 'B')	true	Because (14 >= 5) is true , ('A' < 'B') is true , and true && true is true , the expression evaluates to true .
(24 >= 35) && ('A' < 'B')	false	Because (24 >= 35) is false , ('A' < 'B') is true , and false && true is false , the expression evaluates to false .

TABLE 4-8 The `||` (Or) Operator

Expression1	Expression2	Expression1 Expression2
<code>true</code> (nonzero)	<code>true</code> (nonzero)	<code>true</code> (1)
<code>true</code> (nonzero)	<code>false</code> (0)	<code>true</code> (1)
<code>false</code> (0)	<code>true</code> (nonzero)	<code>true</code> (1)
<code>false</code> (0)	<code>false</code> (0)	<code>false</code> (0)

EXAMPLE 4-4

Expression	Value	Explanation
<code>(14 >= 5) ('A' > 'B')</code>	<code>true</code>	Because <code>(14 >= 5)</code> is <code>true</code> , <code>('A' > 'B')</code> is <code>false</code> , and <code>true false</code> is <code>true</code> , the expression evaluates to <code>true</code> .
<code>(24 >= 35) ('A' > 'B')</code>	<code>false</code>	Because <code>(24 >= 35)</code> is <code>false</code> , <code>('A' > 'B')</code> is <code>false</code> , and <code>false false</code> is <code>false</code> , the expression evaluates to <code>false</code> .
<code>('A' <= 'a') (7 != 7)</code>	<code>true</code>	Because <code>('A' <= 'a')</code> is <code>true</code> , <code>(7 != 7)</code> is <code>false</code> , and <code>true false</code> is <code>true</code> , the expression evaluates to <code>true</code> .



Control Structures



- A computer can proceed:
 - In sequence
 - Selectively (branch) - making a choice
 - Repetitively (iteratively) - looping
- Some statements are executed only if certain conditions are met
- A condition is represented by a logical (Boolean) expression that can be true or false
- A condition is met if it evaluates to true

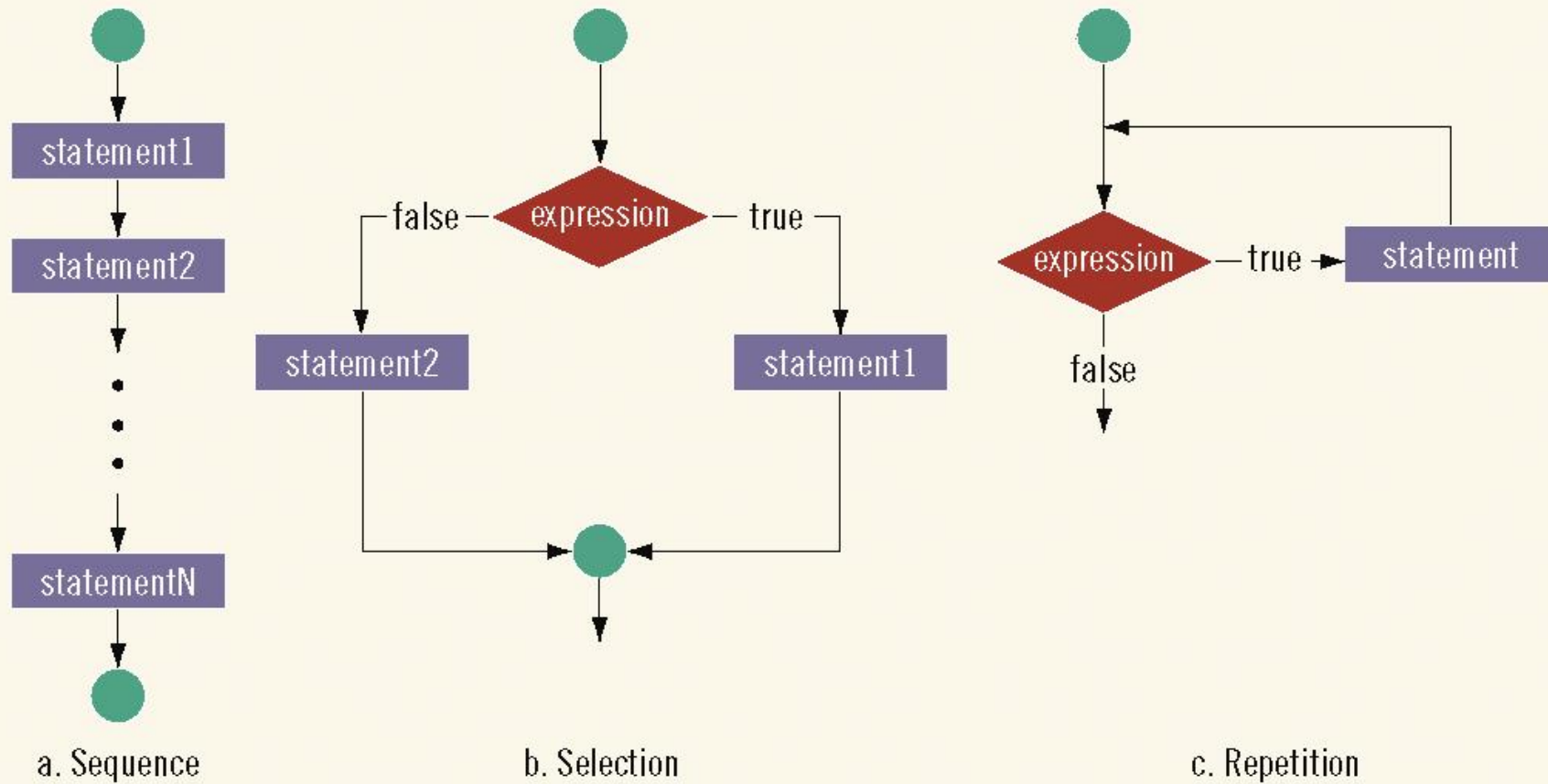


FIGURE 4-1 Flow of execution



Larger of two Numbers

- Write the code to print larger of two numbers.
- **DISPLAY** "Enter the Number1 "
- **READ** Number1
- **DISPLAY** "Enter the Number2 "
- **READ** Number2
- **IF** (Number1 >= Number2)
 - **DISPLAY** "Number1"
- **ELSE**
 - **DISPLAY** "Number2"



Even/odd Number?

- **DISPLAY** "Enter the Number "
- **READ** Number
- **IF**(Number MOD 2 == 0)
 - **DISPLAY** "Number is Even"
- **ELSE**
 - **DISPLAY** "Number is Odd"



Compare two integer values

- **DISPLAY** "Enter two integers: "
- **READ** number1
- **READ** number2
- **if** (number1 == number2)
 - **DISPLAY** "two numbers are equal"
- **else if** (number1 > number2)
 - **DISPLAY** "number1 is greater than number2"
- **else**
 - **DISPLAY** "number1 is less than number2"





Grade

- You have been selected as grader at FAST NUCES; you find it quite tiring to calculate the grade of each student manually.
- You know conditional statements (if/if-else) and decided to automate the system.
- You have to write a pseudocode that will help the programmer to automate the grading system



Grading Criteria


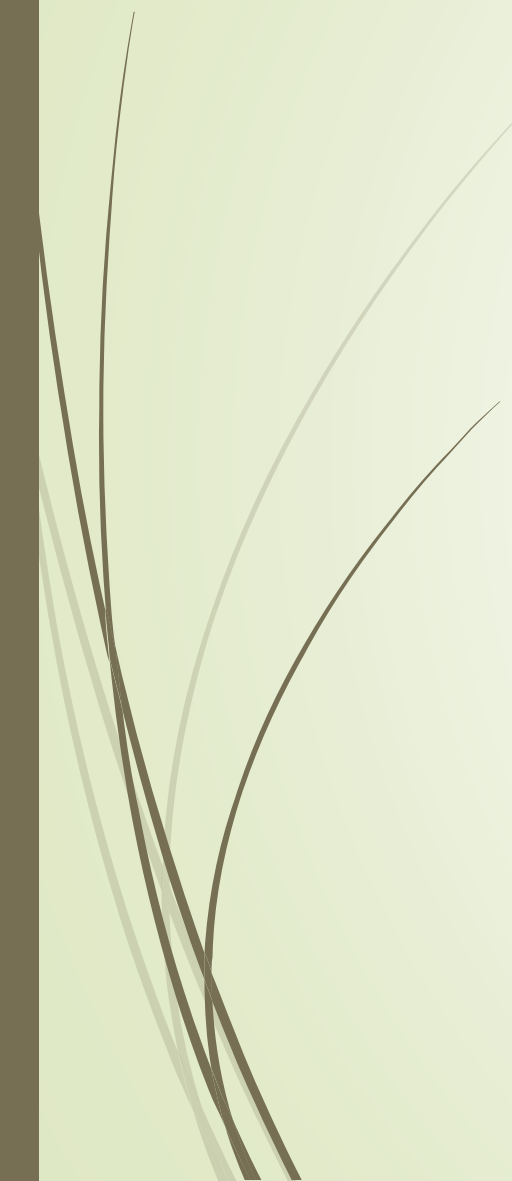
- Students will be assigned with A grade if average marks of student is greater than 80; B grade for students having marks greater than 70; C for marks greater than 60; D for marks greater and equal to 50 otherwise student will be awarded with grade “F”.


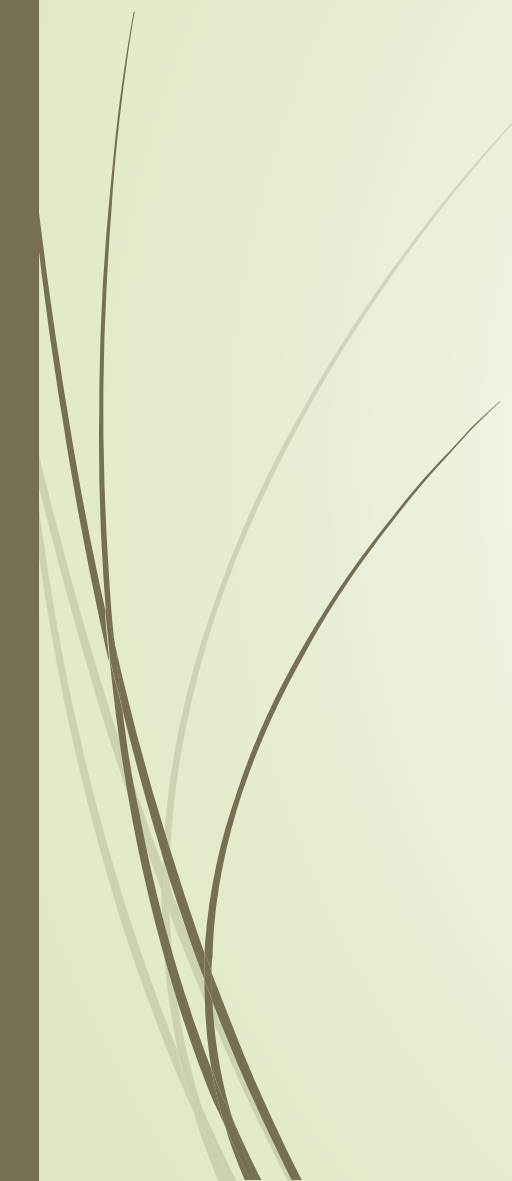
- 
- 
- Display" Enter average marks"
 - Read marks
 - Display " Your Grade is "
 - if (marks > 80)
 - Display "A"
 - else if (marks > 70)
 - Display "B"
 - else if (marks > 60)
 - Display "C"
 - else if (marks > 50)
 - Display "D"
 - else
 - Display "F"



Question

➡ **What Will be the Output? If (marks==80)**

- 
- 
- Display " Enter average marks"
 - Read marks
 - Display " Your Grade is "
 - if (marks \geq 80)
 - Display "A"
 - else if (marks \geq 70)
 - Display "B"
 - else if (marks \geq 60)
 - Display "C"
 - else if (marks \geq 50)
 - Display "D"
 - else
 - Display "F"

- 
- 
- Display " Enter average marks"
 - Read marks
 - Display " Your Grade is "
 - if (marks > 80)
 - Display "A"
 - else if (marks > 70 AND marks <=80)
 - Display "B"
 - else if (marks > 60 AND marks <=70)
 - Display "C"
 - else if (marks > 50 AND marks <=80)
 - Display "D"
 - else
 - Display "F"



Exercise 1

➤ `Int X= 0;`

➤ `Int Z = 1;`

➤ `Cin >>X;`

➤ `Cout << Z;`

What is output of above program if input is

a) X is 3

b) X is 4

c) X is 2

Exercise 1

- `Int X= 0;`
- `Int Z = 1;`
- `Cin >>X;`

- `if (X <4)`
 - `Z = 2;`

- `Cout << Z;`

What is output of above program if input is

- a) X is 3
- b) X is 4
- c) X is 2

Exercise 2

- `Int X= 0;`
- `Int Z = 1;`
- `Cin >> X;`

- `if (X < 4)`
 - `Z = 2;`
- `else`
 - `Z = 4;`

- `Cout << Z;`

What is output of
above program if
input is

- a) X is 1
- b) X is 5
- c) X is 4

Exercise 3

- `Int X= 0;`
- `Int Z = 1;`
- `Cin >> X;`

- `if (X <4)`
 - `Z = 1;`
- `If (X == 3)`
 - `Z = 2;`

- `Cout << Z;`

What is output of above program if input is

- a) X is 3
- b) X is 4
- c) X is 1

One-Way (if) Selection

- The syntax of one-way selection is:

```
if (expression)
```

```
statement
```

- Statement is executed if the value of the expression is `true`
- Statement is bypassed if the value is `false`; program goes to the next statement

EXAMPLE 4-9

```
if (score >= 90)
    grade = 'A';
```

In this code, if the expression `(score >= 90)` evaluates to **true**, the assignment statement, `grade = 'A';`, executes. If the expression evaluates to **false**, the statements (if any) following the **if** structure execute. For example, if the value of `score` is 95, the value assigned to the variable `grade` is 'A'.

EXAMPLE 4-10

The following C++ program finds the absolute value of an integer:

//Program: Absolute value of an integer

```
#include <iostream>

using namespace std;

int main()
{
    int number, temp;

    cout << "Line 1: Please enter an integer: "; //Line 1
    cin >> number;                               //Line 2
    cout << endl;                                //Line 3

    temp = number;                               //Line 4

    if (number < 0)                               //Line 5
        number = -number;                        //Line 6

    cout << "Line 7: The absolute value of "
         << temp << " is " << number << endl; //Line 7

    return 0;
}
```

Sample Run: In this sample run, the user input is shaded.

```
Line 1: Please enter an integer: -6734
Line 7: The absolute value of -6734 is 6734
```

Two-Way (if...else) Selection

- Two-way selection takes the form:

```
if (expression)
```

```
    statement1
```

```
else
```

```
    statement2
```

- If expression is `true`, statement1 is executed otherwise statement2 is executed
- statement1 and statement2 are any C++ statements
- `else` is a reserved word

EXAMPLE 4-13

Consider the following statements:

```
if (hours > 40.0)           //Line 1
    wages = 40.0 * rate +
        1.5 * rate * (hours - 40.0); //Line 2
else                         //Line 3
    wages = hours * rate;    //Line 4
```

if the value of the variable `hours` is greater than `40.0`, then the `wages` include overtime payment. Suppose that `hours` is `50`. The expression in the `if` statement, in Line 1, evaluates to `true`, so the statement in Line 2 executes. On the other hand, if `hours` is `30`, or any number less than or equal to `40`, the expression in the `if` statement, in Line 1, evaluates to `false`. In this case, the program skips the statement in Line 2 and executes the statement in Line 4—that is, the statement following the reserved word `else` executes.

Indentation

```
if (employed == 'Y')
{
if (recentGrad == 'Y') // Nested if
{
cout << "You qualify for the special ";
cout << "interest rate.\n";
}
else // Not a recent grad, but employed
{
cout << "You must have graduated from ";
cout << "college in the past two\n";
cout << "years to qualify.\n";
}
}
else // Not employed
{
cout << "You must be employed to qualify.\n";
}
```

*Don't write code
like this!*


```
if (employed == 'Y')
{
    if (recentGrad == 'Y') // Nested if
    {
        cout << "You qualify for the special ";
        cout << "interest rate.\n";
    }
    else // Not a recent grad, but employed
    {
        cout << "You must have graduated from ";
        cout << "college in the past two\n";
        cout << "years to qualify.\n";
    }
}
else // Not employed
{
    cout << "You must be employed to qualify.\n";
}
```

This if and else go together.

This if and else go together.



References



1. C++ Programming: From Problem Analysis to Program Design, Third Edition
2. <https://www.just.edu.jo/~yahya-t/cs115/>