

THIRD EDITION

A First Book of C++

From Here to There

CHAPTER 4

Selection

Objectives

You should be able to describe:

- Relational Expressions
- The `if-else` Statement
- Nested `if` Statements
- The `switch` Statement
- Common Programming Errors

Selection

- **Flow of Control:** the order in which a program's statements are executed
 - Normal flow is sequential
- Selection and Repetition Statements allow programmer to alter normal flow
- **Selection:** selects a particular statement to be executed next
 - Selection is from a well-defined set
- **Repetition:** allows a set of statements to be repeated

Relational Expressions

- All computers are able to compare numbers
 - Can be used to create an intelligence-like facility
- **Relational Expressions:** expressions used to compare operands
 - **Format:** a relational operator connecting two variable and/or constant operands
 - Examples of valid relational expressions:
`Age > 40 length <= 50 flag == done`

Relational Expressions (continued)

TABLE 4.1 *Relational Operators in C++*

Relational Operator	Meaning	Example
<	less than	age < 30
>	greater than	height > 6.2
<=	less than or equal to	taxable <= 20000
>=	greater than or equal to	temp >= 98.6
==	equal to	grade == 100
!=	not equal to	number != 250

Relational Expressions (continued)

- **Relational Expressions** (conditions):
 - Are evaluated to yield a numerical result
 - Condition that is true evaluates to 1
 - Condition that is false evaluates to 0
- **Example:**
 - The relationship $2.0 > 3.3$ is always false, therefore the expression has a value of 0

Logical Operators

- More complex conditions can be created using logical operations `AND`, `OR`, and `NOT`
 - Represented by the symbols: `&&`, `||`, `!`
- **AND Operator, `&&`:**
 - Used with 2 simple expressions
 - **Example:** `(age > 40) && (term < 10)`
 - Compound condition is true (has value of 1) only if `age > 40` and `term < 10`

Logical Operators (continued)

- **OR Operator, || :**
 - Used with 2 simple expressions
 - **Example:** `(age > 40) || (term < 10)`
 - Compound condition is true if `age > 40` or if `term < 10` or if both conditions are true
- **NOT Operator, ! :**
 - Changes an expression to its opposite state
 - If `expressionA` is true, then `!expressionA` is false

Logical Operators (continued)

TABLE 4.2 *Precedence of Relational and Logical Operators*

Operator	Associativity
! unary - ++ --	right to left
* / %	left to right
+ -	left to right
< <= > >=	left to right
== !=	left to right
&&	left to right
	left to right
= += -= *= /=	right to left

A Numerical Accuracy Problem

- Avoid testing equality of single and double-precision values and variables using `==` operator
 - Tests fail because many decimals cannot be represented accurately in binary
- **For real operands:**
 - The expression
$$\text{operand_1} == \text{operand_2}$$
should be replaced by
$$\text{abs}(\text{operand_1} - \text{operand_2}) < \text{EPSILON}$$
 - If this expression is true for very small `EPSILON`, then the two operands are considered equal

The `if-else` Statement

- Selects a sequence of one or more instructions based on the results of a comparison

- **General form:**

```
if (expression)      <- no semicolon here
    statement1;
else                  <- no semicolon here
    statement2;
```

- If the value of `expression` is true, `statement1` is executed
- If the value is false, `statement2` is executed

The `if-else` Statement (continued)



Program 4.1

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float taxable, taxes;

    cout << "Please type in the taxable income: ";
    cin >> taxable;

    if (taxable <= 20000.0)
        taxes = 0.02 * taxable;
    else
        taxes = 0.025 * (taxable - 20000.0) + 400.0;

    cout << setiosflags(ios::fixed)
         << setprecision(2)
         << "Taxes are $ " << taxes << endl;

    return 0;
}
```

The `if-else` Statement (continued)

- Program 4.1 run twice with different input data

- Result 1:

- Please type in the taxable income:

- 10000

- Taxes are \$ 200.00

- Result 2:

- Please type in the taxable income:

- 30000

- Taxes are \$ 650.00

Compound Statements

FIGURE 4.2 *A Compound Statement Consists of Individual Statements Enclosed Within Braces*

```
if (expression)
{
    statement1;    // as many statements as necessary
    statement2;    // can be put within the braces
    statement3;    // each statement must end with a ;
}
else
{
    statement4;
    statement5;
    .
    .
    .
    statementn;
}
```

Compound Statements (continued)



Program 4.2

```
#include <iostream>
#include <iomanip>
using namespace std;

// a temperature conversion program
int main()
{
    char tempType;
    double temp, fahrenheit, celsius;

    cout << "Enter the temperature to be converted: ";
    cin >> temp;
    cout << "Enter an f if the temperature is in Fahrenheit";
    cout << "\n or a c if the temperature is in Celsius: ";
    cin >> tempType;

    // set output formats
    cout << setiosflags (ios::fixed)
         << setiosflags (ios::showpoint)
         << setprecision(2);
```

Compound Statements (continued)

Program 4.2 (continued):

```
    if (tempType == 'f')
    {
        celsius = (5.0 / 9.0) * (temp - 32.0);
        cout << "\nThe equivalent Celsius temperature is " << celsius
              << endl;
    }
    else
    {
        fahrenheit = (9.0 / 5.0) * temp + 32.0;
        cout << "\nThe equivalent Fahrenheit temperature is " << fahrenheit
              << endl;
    }

    return 0;
}
```

Compound Statements (continued)

Output of Program 4.2

```
Enter the temperature to be converted:
```

```
212
```

```
Enter an f if the temperature is in  
Fahrenheit or a c if the temperature  
is in Celsius: f
```

```
The equivalent Celsius temperature is
```

```
100.00
```

Compound Statements (continued)

- **bool**: a C++ built-in Boolean Data Type
 - Two Boolean values: `true` and `false`
- Values of `true` and `false`: integer values 1 and 0 respectively
- To see Boolean values displayed as `true` and `false` insert the manipulator `boolalpha` into the `cout` stream prior to displaying Boolean values
- Applying prefix or postfix increment (`++`) to a `bool` variable sets its value to `true`

Block Scope

- **Block of Code:** All statements contained within a compound statement
- Any variable declared within a block has meaning only between its declaration and the closing braces of the block
- Example with two blocks of code

Block Scope (continued)

```
{ // start of outer block
  int a = 25;
  int b = 17;
  cout << "The value of a is " << a << " and b is "
  << b << endl;
  { // start of inner block
    double a = 46.25;
    int c = 10;
    cout << "a is now " << a
          << " b is now " << b
          << " and c is " << c << endl;
  } // end of inner block
  cout << "a is now " << a << " and b is " << b <<
  endl;
} // end of outer block
```

Block Scope (continued)

- **Output of Block Scope example:**

```
The value of a is 25 and b is 17  
a is now 46.25 b is now 17 and c is 10  
a is now 25 and b is 17
```

Block Scope (continued)

- **Common programming practice:** place opening brace of a compound statement on the same line as `if` and `else` statements

```
if (tempType == 'f') {  
    celcius = (5.0 / 9.0) * (temp - 32);  
    cout << "\nThe equivalent Celsius temperature is"  
        << celcius << endl;  
}
```

Block Scope (continued)

- **The traditional format:**

```
If (tempType == 'f')
{
    celsius = (5.0 / 9.0) * (temp - 32);
    cout << "\nThe equivalent Celsius
    temperature is "
        << celsius << endl;
}
```

One-Way Selection

- A modification of `if-else` that omits `else` part
 - `if` statement takes the form:

```
if (expression)  
    else;
```
- Modified form called a one-way statement
 - The statement following `if (expression)` is executed only if the expression is true
 - The statement may be a compound statement

One-Way Selection (continued)



Program 4.3

```
#include <iostream>
using namespace std;

int main()
{

    const double LIMIT = 3000.0;
    int idNum;
    double miles;

    cout << "Please type in car number and mileage: ";
    cin  >> idNum >> miles;

    if(miles > LIMIT)
        cout << " Car " << idNum << " is over the limit." << endl;

    cout << "End of program output." << endl;

    return 0;
}
```

One-Way Selection (continued)

- Program 4.3 run twice with different input data

- Result 1:

```
Please type in car number and mileage:  
256 3562.8  
Car 256 is over the limit.  
End of program output.
```

- Result 2:

```
Please type in car number and mileage:  
23 2562.8  
End of program output.
```

Problems Associated with the `if-else` Statement

- **Most common problems:**
 - Misunderstanding what an expression is
 - Using the assignment operator, `=`, in place of the relational operator, `==`
- **Example:**
 - Initialize `age = 18`
 - The expression `(age = 30)` sets `age` to 30
 - Does not compare `age` to 30
 - Has a value of 30 (true)
 - Produces invalid results if used in `if-else` statement

Problems Associated with the `if-else` Statement (continued)

- Example continued:
 - The expression `(age == 30)` compares `age` to `30`
 - has a value of 0 (false)
 - This expression will produce a valid test in an `if-else` statement

Nested `if` Statements

- `if-else` statement can contain simple or compound statements
 - Another `if-else` statement can be included

- **Example:**

```
If (hours < 9)
{
    if (hours > 6)
        cout << "snap";
}
else
    cout << "pop";
```

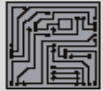
The `if-else` Chain

- **Format:**

```
if (expression_1)
    statement1;
else
    if (expression_2)
        statement2;
    else
        statement3;
```

- Chain can be extended indefinitely by making last statement another `if-else` statement

The `if-else` Chain (continued)



Program 4.4

```
#include <iostream>
using namespace std;

int main()
{
    char marcode;

    cout << "Enter a marital code: ";
    cin  >> marcode;

    if (marcode == 'M')
        cout << "Individual is married." << endl;
    else if (marcode == 'S')
        cout << "Individual is single." << endl;
    else if (marcode == 'D')
        cout << "Individual is divorced." << endl;
    else if (marcode == 'W')
        cout << "Individual is widowed." << endl;
    else
        cout << "An invalid code was entered." << endl;

    return 0;
}
```

The `switch` Statement

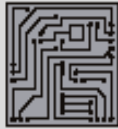
- **Format:**

```
switch (expression)
{
    // start of compound statement
    case value_1:    <- terminated with a colon
        statement1;
        statement2;
        break;
    case value_2:    <- terminated with a colon
        statementm;
        break;
    default:          <- terminated with a colon
        statementaa;
}
    // end of switch and compound
    // statement
```

The `switch` Statement (continued)

- **Four new keywords used:**
 - `switch`, `case`, `default` and `break`
- **Function:**
 - Expression following `switch` is evaluated
 - Must evaluate to an integer result
 - Result compared sequentially to alternative `case` values until a match found
 - Statements following matched `case` are executed
 - When `break` statement reached, `switch` terminates
 - If no match found, default statement block is executed

The `switch` Statement (continued)



Program 4.6

```
#include <iostream>
using namespace std;
int main()
{
    int opselect;
    double fnum, snum;

    cout << "Please type in two numbers: ";
    cin  >> fnum >> snum;
    cout << "Enter a select code: ";
    cout << "\n      1 for addition";
    cout << "\n      2 for multiplication";
    cout << "\n      3 for division : ";
    cin  >> opselect;
```

The `switch` Statement (continued)

Program 4.6 (continued):

```
switch (opselect)
{
    case 1:
        cout << "The sum of the numbers entered is " << fnum+snum << endl;
        break;
    case 2:
        cout << "The product of the numbers entered is " << fnum*snum << endl;
        break;
    case 3:
        cout << "The first number divided by the second is " << fnum/snum << endl;
        break;
}    // end of switch

return 0;
}    // end of main()
```

The `switch` Statement (continued)

Program 4.6 results:

```
Please type in two numbers: 12 3
```

```
Enter a select code:
```

```
    1 for addition
```

```
    2 for multiplication
```

```
    3 for division : 2
```

```
The product of the numbers entered is 36
```

Common Programming Errors

- Using the assignment operator , =, in place of the relational operator, ==
- Assuming that the `if-else` statement is selecting an incorrect choice when the problem is really the values being tested
- Using nested `if` statements without including braces to clearly indicate the desired structure

Summary

- Relational Expressions (conditions):
 - Are used to compare operands
 - A condition that is true has a value of 1
 - A condition that is false has a value of 0
- More complex conditions can be constructed from relational expressions using C++'s logical operators, `&&` (AND), `||` (OR), and `!` (NOT)
- `if-else` statements select between two alternative statements based on the value of an expression

Summary (continued)

- `if-else` statements can contain other `if-else` statements
 - If braces are not used, each `else` statement is associated with the closest unpaired `if`
- **`if-else Chain`**: a multi-way selection statement
 - Each `else` statement (except for the final `else`) is another `if-else` statement
- **`Compound Statement`**: any number of individual statements enclosed within braces

Summary (continued)

- Variables have meaning only within the block where they are declared
 - Includes any inner blocks
- **switch Statement:** multiway selection statement
 - The value of an integer expression is compared to a sequence of integer or character constants or constant expressions
 - Program execution transferred to first matching case
 - Execution continues until optional break statement is encountered