Making Decisions

CS 1: Problem Solving & Program Design Using C++

Objectives

- Relate to relational operators
- Logically look at logical operators
- Discover if/else statements
- Find out about compound statements and block scope
- Switch it up to the switch statement
- Cover more common programming errors involving if/else and switch

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 (2)

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	Score == 100
!=	Not equal to	Number != 250

Relational Expressions (3)

- 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 symbols
  - AND: &&
  - OR: ||
  - NOT: !

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

X	Y	$X \wedge Y$
F	F	F
F	T	F
T	F	F
T	T	T

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

X	Y	$X \vee Y$
F	F	F
F	T	T
T	F	T
T	T	T

NOT (!) Operator

- Changes an expression to its opposite state
- If expressionA is true, then !expressionA is false

X	¬X
F	T
T	F

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

    operand_1 == operand_2
  - should be replaced by

    abs(operand_1 – operand_2) < 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 Example

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
		float taxable, taxes;

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

		if (taxable <= 20000) {
				taxes = 0.20 * taxable;
		}
		else {
				taxes = 0.25 * (taxable - 20000) + 4000.0;
		}

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

		return 0;
}
```

The if-else Statement Example Sample Runs

- Result 1:

```
Please type in the taxable income: 10000
Taxes are $ 2000.00
```

- Result 2:

```
Please type in the taxable income:  30000
Taxes are $ 6500.00
```

Compound Statements

```
if (statement)
{
     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 Example

```
#include <iostream>
#include <iomanip>

using namespace std;

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;

        cout << setiosflags(ios::fixed)
                << setiosflags(ios::showpoint)
                << setprecision(2);
```

Compound Statements Example (2)

```
        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;
}
```

The Compound Statements Example Sample Run

```
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
```

More on Compound Statements

- bool: a C++ built-in Boolean Data Type
- Two Boolean values
  - True (value of 1)
  - False (value of 0)
- 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

Block Scope Example

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

int main()
{
        // 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;

        return 0;
}   // end of outer block
```

Block Scope Example Output

```
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 Programming Practice

- Place opening brace of a compound statement on the same line as if and else statements

```
  if (tempType == ‘f’) {
      celsius = (5.0 / 9.0)  *  (temp – 32.0);
      cout << “\nThe equivalent Celsius temperature is “ << celsius << 
endl;
  }
```

Block Scope Programming Practice (2)

- Traditional format

```
    if (tempType == ‘f’)
    {
        celsius = (5.0 / 9.0)  *  (temp – 32.0);
        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)
                statement;
```

- 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 Example

```
#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 Example Sample Runs

- 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 if $(\text{age} = 30)$ 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

- On the other hand…
  - The expression `if (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 Example

```
#include <iostream>

using namespace std;

int main()
{
        char marcode;

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

        if (marcode == 'M')
                cout << "Game over." << endl;
        else if (marcode == 'S')
                cout << "Single and ready to mingle." << endl;
        else if (marcode == 'D')
                cout << "My money is their money." << endl;
        else if (marcode == 'W')
                cout << "Starting over again." << 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;
                        break;
}                                   // end of switch and compound
                                    // statement
```

The switch Statement (2)

- 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 Example

```
#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 Example (2)

```
        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
number is " << fnum / snum << endl;
                break;
        }
        return 0;
}
```

The switch Statement Sample Run

```
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)
  - ! (NOT)
- if-else statements select between two alternative statements based on the value of an expression

Summary (2)

- 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 (3)

- 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