RELATIONAL OPERATORS

- Used to compare numbers to determine relative order
- Operators:

  >       Greater than
  <       Less than
  >=      Greater than or equal to
  <=      Less than or equal to
  ==      Equal to
  !=      Not equal to
RELATIONAL OPERATORS

- Boolean expressions – true or false
- Examples:
  
  12 > 5 is true
  7 <= 5 is false
  
  if x is 10, then
  x == 10 is true,
  x != 8 is true, and
  x == 8 is false
  
- Can be assigned to a variable:
  result = x <= y;
- Assigns 0 for false, 1 for true
- Do not confuse = and ==

THE if STATEMENT

- Allows statements to be conditionally executed or skipped over
- Models the way we mentally evaluate situations:
  - "If it is raining, take an umbrella."
  - "If it is cold outside, wear a coat."
THE \textbf{if} STATEMENT
FLOWCHART FOR EVALUATING A DECISION

\begin{center}
\begin{tikzpicture}
  \node (q0) [evaluate decision] {Is it cold outside?};
  \node (q1) [below of=q0, anchor=west, xshift=-1.5cm] {Wear a coat.};
  \node (q2) [below of=q0, anchor=east, xshift=1.5cm] {Wear a hat.};
  \node (q3) [below of=q2, anchor=west, xshift=-1.5cm] {Wear gloves.};

  \path[->]
  (q0) edge [loop above] node {Yes} (q1)
  (q0) edge [loop below] node {No} (q2)
  (q2) edge [loop below] node {No} (q3)

\end{tikzpicture}
\end{center}
THE if STATEMENT

GENERAL FORMAT

    if (expression)
        statement;

- If the expression is true, then statement is executed.
- If the expression is false, then statement is skipped.

THE if STATEMENT

IN PROGRAM

Program 4-2

```cpp
    // This program averages three test scores
    #include <iostream>
    #include <iomanip>
    using namespace std;

    int main()
    {
        int score1, score2, score3;  // To hold three test scores
        double average;               // To hold the average score
```
THE if STATEMENT
IN PROGRAM

Program 4-2 (continued)

```cpp
// Get the three test scores.
11 cout << "Enter 3 test scores and I will average them: ");
12 cin >> score1 >> score2 >> score3;
13 // Calculate and display the average score.
14 average = (score1 + score2 + score3) / 3.0;
15 cout << fixed << showpoint << setprecision(1);
16 cout << "Your average is: " << average << endl;
17 // If the average is greater than 95, congratulate the user.
18 if (average > 95)
19    cout << "Congratulations! That's a high score!\n";
20 return 0;
21 }
```

**Program Output with Example input Shown in bold**
Enter 3 test scores and I will average them: 80 90 70 [Enter]
Your average is 80.0

**Program Output with Other Example Input Shown in Bold**
Enter 3 test scores and I will average them: 100 100 100 [Enter]
Your average is 100.0
Congratulations! That's a high score!

THE if STATEMENT
FLOWCHART – LINE 21,22
THE if STATEMENT

NOTES

• Do not place ; after (expression)
• Place statement; on a separate line after (expression), indented:

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

• Be careful testing floats and doubles for equality
• 0 is false; any other value is true

EXPANDING THE if STATEMENT

NOTES

• To execute more than one statement as part of an if statement, enclose them in {}:

```
if (score > 90)
{
    grade = 'A';
    cout << "Good Job!\n";
}
```

• {} creates a block of code
THE if/else STATEMENT

• Provides two possible paths of execution
• Performs one statement or block if the expression is true, otherwise performs another statement or block.

• General Format:

```
if (expression)
    statement1; // or block
else
    statement2; // or block
```

THE if/else STATEMENT

WHAT HAPPENS

• To evaluate:

```
if (expression)
    statement1;
else
    statement2;
```

• If the expression is true, then statement1 is executed and statement2 is skipped.
• If the expression is false, then statement1 is skipped and statement2 is executed.
**THE if/else STATEMENT**

**MODULUS OPERATOR**

**Program 4.8**

```cpp
1 // This program uses the modulus operator to determine
2 // if a number is odd or even. If the number is evenly divisible
3 // by 2, it is an even number. A remainder indicates it is odd.
4 #include <iostream>
5 using namespace std;
6
7 int main()
8 {
9    int number;
10    cout << "Enter an integer and I will tell you if it\n";
11    cout << "is odd or even. ";
12    cin >> number;
13    if (number % 2 == 0)
14        cout << number << " is even.\n";
15    else
16        cout << number << " is odd.\n";
17    return 0;
18 }
```

**Program Output with Example Input Shown in Bold**

Enter an integer and I will tell you if it is odd or even. 17 [Enter]
17 is odd.

---

**THE if/else STATEMENT**

**FLOW CHART**

![Flowchart](image)

- True: \( \text{number} \mod 2 == 0 \)
  - Indicate that the number is even.
- False: \( \text{number} \mod 2 \neq 0 \)
  - Indicate that the number is odd.
THE if/else STATEMENT
TESTING THE DIVISOR

Program 4-9

1 // This program asks the user for two numbers, num1 and num2.
2 // num1 is divided by num2 and the result is displayed.
3 // Before the division operation, however, num2 is tested
4 // for the value 0. If it contains 0, the division does not
5 // take place.
6 #include <iostream>
7 using namespace std;
8
9 int main()
10 {
11    double num1, num2, quotient;
12
13 // Get the first number.
14    cout << "Enter a number: ";
15    cin >> num1;
16
17 // Get the second number.
18    cout << "Enter another number: ";
19    cin >> num2;
20
21 // If num2 is not zero, perform the division.
22    if (num2 != 0)
23        {
24        cout << "Division by zero is not possible.
25        cout << "Please run the program again and enter a number other than zero.\n";
26        cout << "a number other than zero.\n";
27        }
28    else
29    {
30        quotient = num1 / num2;
31        cout << "The quotient of " << num1 << " divided by ";
32        cout<< num2 << " is " << quotient << ",\n";
33    }
34    return 0;
35 }

Program Output with Example Input Shown in Bold
(When the user enters 0 for num2)
Enter a number: 10 [Enter]
Enter another number: 0 [Enter]
Division by zero is not possible.
Please run the program again and enter a number other than zero.
NESTED if STATEMENT

- An if statement that is nested inside another if statement
- Nested if statements can be used to test more than one condition

```
20 // Determine the user's loan qualifications.
21 if (employed == 'Y')
22 {
23     if (recentGrad == 'Y') //Nested if
24     {
25         cout << "You qualify for the special ";
26         cout << "interest rate.\n";
27     }
28 }
```
NESTED if STATEMENT

ANOTHER EXAMPLE

```cpp
// Determine the user's loan qualifications.
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";
    }
}
```

NESTED if STATEMENT

USE PROPER INDENTATION

```cpp
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";
    }
```
THE if/else if STATEMENT

- Tests a series of conditions until one is found to be true
- Often simpler than using nested if/else statements
- Can be used to model thought processes such as:

  "If it is raining, take an umbrella, else, if it is windy, take a hat, else, take sunglasses"

```cpp
if (expression)
    statement1; // or block
else if (expression)
    statement2; // or block
    // other else ifs
else if (expression)
    statementn; // or block
```

THE if/else if STATEMENT

THE PROGRAM

```cpp
// Determine the letter grade.
if (testScore >= A_SCORE)
    cout << "Your grade is A.\n";
else if (testScore >= B_SCORE)
    cout << "Your grade is B.\n";
else if (testScore >= C_SCORE)
    cout << "Your grade is C.\n";
else if (testScore >= D_SCORE)
    cout << "Your grade is D.\n";
else
    cout << "Your grade is F.\n";
```
**THE `if/else if` STATEMENT**

**THE PROGRAM**

- The trailing `else` clause is optional, but it is best used to catch errors.

```cpp
21    // Determine the letter grade.
22    if (testScore >= A_SCORE)  
23        cout << "Your grade is A.\n";
24    else if (testScore >= B_SCORE)  
25        cout << "Your grade is B.\n";
26    else if (testScore >= C_SCORE)  
27        cout << "Your grade is C.\n";
28    else if (testScore >= D_SCORE)  
29        cout << "Your grade is D.\n";
30    else if (testScore >= 0)  
31        cout << "Your grade is F.\n";
32    else  
33        cout << "Invalid test score.\n";
```

**FLAGS**

- Variable that signals a condition
  - Usually implemented as a bool variable
  - Can also be an integer
    - The value 0 is considered false
    - Any nonzero value is considered true
  - As with other variables in functions, must be assigned an initial value before it is used
LOGICAL OPERATORS

- Used to create relational expressions from other relational expressions
- Operators, meaning, and explanation:

<table>
<thead>
<tr>
<th>&amp; &amp;</th>
<th>AND</th>
<th>New relational expression is true if both expressions are true</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOT</td>
</tr>
</tbody>
</table>

```java
int x = 12, y = 5, z = -4;

(x > y) && (y > z)  // true
(x > y) && (z > y)  // false
(x <= z) || (y == z)  // false
(x <= z) || (y != z)  // true
!(x >= z)  // false
```
LOGICAL OPERATORS
IN PROGRAM

```cpp
21    // Determine the user's loan qualifications.
22    if (employed == 'Y' && recentGrad == 'Y')
23       {
24       cout << "You qualify for the special "
25       << "interest rate.\n";
26       }
27    else
28       {
29       cout << "You must be employed and have\n"
30       << "graduated from college in the\n"
31       << "past two years to qualify.\n";
32       }
```

LOGICAL OPERATORS
LOGICAL || OPERATOR

```cpp
23    // Determine the user's loan qualifications.
24    if (income >= MIN_INCOME || years > MIN_YEARS)
25       cout << "You qualify.\n";
26    else
27       {
28       cout << "You must earn at least $"  
29       << MIN_INCOME << " or have been "
30       << "employed more than " << MIN_YEARS
31       << " years.\n";
32       }
```
LOGICAL OPERATORS

LOGICAL ! OPERATOR

```cpp
23    // Determine the user's loan qualifications.
24    if (!(income >= MIN_INCOME || years > MIN_YEARS))
25    {
26        cout << "You must earn at least $";
27        << MIN_INCOME << " or have been "
28        << "employed more than " << MIN_YEARS
29        << " years.\n";
30    }
31    else
32    cout << "You qualify.\n";
```

LOGICAL OPERATORS

NOTES

• ! has highest precedence, followed by &&, then ||

• If the value of an expression can be determined by evaluating just the sub-expression on left side of a logical operator, then the sub-expression on the right side will not be evaluated (short circuit evaluation)
LOGICAL OPERATORS
CHECKING NUMERIC RANGES

• Used to test to see if a value falls inside a range:
  ```
  if (grade >= 0 && grade <= 100)
  cout << "Valid grade";
  ```

• Can also test to see if value falls outside of range:
  ```
  if (grade <= 0 || grade >= 100)
  cout << "Invalid grade";
  ```

• Cannot use mathematical notation:
  ```
  if (0 <= grade <= 100) //doesn't work!
  ```

MENUS
MENU-DRIVEN PROGRAM ORGANIZATION

• Menu-driven program: program execution controlled by user selecting from a list of actions
• Menu: list of choices on the screen
• Menus can be implemented using if/else if statements

• Display list of numbered or lettered choices for actions
• Prompt user to make selection
• Test user selection in expression
  • if a match, then execute code for action
  • if not, then go on to next expression
VALIDATING USER INPUT

- **Input validation**: inspecting input data to determine whether it is acceptable

- Bad output will be produced from bad input

- Can perform various tests:
  - Range
  - Reasonableness
  - Valid menu choice
  - Divide by zero

```cpp
int testScore; // To hold a numeric test score

// Get the numeric test score.
cout << "Enter your numeric test score and I will\n" << "tell you the letter grade you earned: ";
cin >> testScore;

// Validate the input and determine the grade.
if (testScore >= MIN_SCORE && testScore <= MAX_SCORE) {
  // Determine the letter grade.
  if (testScore >= A_SCORE)
    cout << "Your grade is A.\n";
  else if (testScore >= B_SCORE)
    cout << "Your grade is B.\n";
  else if (testScore >= C_SCORE)
    cout << "Your grade is C.\n";
  else if (testScore >= D_SCORE)
    cout << "Your grade is D.\n";
  else
    cout << "Your grade is F.\n";
} else
{
  // An invalid score was entered.
  cout << "That is an invalid score. Run the program\n" << "again and enter a value in the range of\n" << MIN_SCORE << " through " << MAX_SCORE << ".\n";
}
```
COMPARING CHARACTERS AND STRINGS

• Characters are compared using their ASCII values

• 'A' < 'B'
  • The ASCII value of 'A' (65) is less than the ASCII value of 'B'(66)

• '1' < '2'
  • The ASCII value of '1' (49) is less than the ASCII value of '2' (50)

• Lowercase letters have higher ASCII codes than uppercase letters, so 'a' > 'Z'

```cpp
10  // Get a character from the user.
11  cout << "Enter a digit or a letter: ";
12  ch = cin.get();
13
14  // Determine what the user entered.
15  if (ch >= '0' && ch <= '9')
16      cout << "You entered a digit.\n";
17  else if (ch >= 'A' && ch <= 'Z')
18      cout << "You entered an uppercase letter.\n";
19  else if (ch >= 'a' && ch <= 'z')
20      cout << "You entered a lowercase letter.\n";
21  else
22      cout << "That is not a digit or a letter.\n";
```
COMPARING CHARACTERS AND STRINGS

STRING OBJECTS

- Like characters, strings are compared using their ASCII values

    string name1 = "Mary";
    string name2 = "Mark",

    name1 > name2 // true
    name1 <= name2 // false
    name1 != name2 // true

    name1 < "Mary Jane" // true

    The characters in each string must match before they are equal

COMPARING CHARACTERS AND STRINGS

IN PROGRAM

```c++
    // Determine and display the correct price
    if (partNum == "S-29A")
        cout << "The price is $" << PRICE_A << endl;
    else if (partNum == "S-29B")
        cout << "The price is $" << PRICE_B << endl;
    else
        cout << partNum << " is not a valid part number.\n";
```
THE CONDITIONAL OPERATOR

- Can use to create short if/else statements
- Format: expr ? expr : expr;

```
x<0  ?  y=10  :  z=20;
```

First Expression:
Expression to be tested

2nd Expression:
Executes if first expression is true

3rd Expression:
Executes if the first expression is false

THE CONDITIONAL OPERATOR

- The value of a conditional expression is
  - The value of the second expression if the first expression is true
  - The value of the third expression if the first expression is false

- Parentheses () may be needed in an expression due to precedence of conditional operator
THE CONDITIONAL OPERATOR

```cpp
1 // This program calculates a consultant's charges at $50
2 // per hour, for a minimum of 5 hours. The ?: operator
3 // adjusts hours to 5 if less than 5 hours were worked.
4 #include <iostream>
5 #include <iomanip>
6 using namespace std;
7
8 int main()
9 {
10   const double PAY_RATE = 50.0; // Hourly pay rate
11   const int MIN_HOURS = 5; // Minimum billable hours
12   double hours, // Hours worked
13       charges; // Total charges
14
15   // Get the hours worked.
16   cout << "How many hours were worked? ";
17   cin >> hours;
18
19   // Determine the hours to charge for.
20   hours = hours < MIN_HOURS ? MIN_HOURS : hours;
21
22   // Calculate and display the charges.
23   charges = PAY_RATE * hours;
24   cout << fixed << showpoint << setprecision(2)
25       << "The charges are $" << charges << endl;
26   return 0;
27 }
```

THE SWITCH STATEMENT

- Used to select among statements from several alternatives
- In some cases, can be used instead of if/else if statements

```cpp
switch (expression) //integer
{
  case exp1: statement1;
  case exp2: statement2;
  ...
  case expn: statementn;
  default: statementn+1;
}
```
THE SWITCH STATEMENT

IN PROGRAM

Program 4.23

```c
1 // The switch statement in this program tells the user something
2 // he or she already knows: the data just entered
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     char choice;
9
10    cout << "Enter A, B, or C: ";
11    cin >> choice;
12    switch (choice)
13    {
14        case 'A': cout << "You entered A."; break;
15        case 'B': cout << "You entered B."; break;
16        case 'C': cout << "You entered C."; break;
17        default: cout << "You did not enter A, B, or C.";
18    }
19    return 0;
20 }
```

Program Output with Example Input Shown in Bold
Enter A, B, or C [Enter]
You entered B.

Program Output with Example Input Shown in Bold
Enter A, B, or C [Enter]
You did not enter A, B, or C

THE SWITCH STATEMENT

REQUIREMENTS

1. **expression** must be an integer variable or an expression that evaluates to an integer value

2. **exp1** through **expn** must be constant integer expressions or literals, and must be unique in the **switch** statement

3. **default** is optional but recommended
THE SWITCH STATEMENT

HOW IT WORKS

1. expression is evaluated

2. The value of expression is compared against exp1 through expn.

3. If expression matches value expi, the program branches to the statement following expi and continues to the end of the switch.

4. If no matching value is found, the program branches to the statement after default:

THE BREAK STATEMENT

- Used to exit a switch statement.
- If it is left out, the program "falls through" the remaining statements in the switch statement.

Program 4-25

```cpp
1  // This program is carefully constructed to use the "fall through"
2  // feature of the switch statement.
3  #include <iostream>
4  using namespace std;
5  
6  int main()
7  { 
8      int modelNum; // Model number
9      // Get a model number from the user.
10     cout << "Our TVs come in three models:\n";
11     cout << "The 100, 200, and 300. Which do you want? ";
12     cin >> modelNum;
13     // Display the model's features.
14     cout << "That model has the following features:\n";
15     switch (modelNum)
16     { 
17         case 300: cout << "*Picture-in-a-picture.\n";
18         case 200: cout << "*Stereo sound.\n";
19         case 100: cout << "*Remote control.\n";
20         break;
21         default: cout << "You can only choose the 100,*;";
22             cout << "200, or 300.\n";
23     } 
24     return 0;
25  }
```
THE BREAK STATEMENT
BREAK AND DEFAULT STATEMENTS IN PROGRAM

Program Output with Example Input Shown in Bold
Our TVs come in three models: The 100, 200, and 300. Which do you want? 100 [Enter]
That model has the following features:
Remote control.

Program Output with Example Input Shown in Bold
Our TVs come in three models: The 100, 200, and 300. Which do you want? 200 [Enter]
That model has the following features:
Stereo sound.
Remote control.

Program Output with Example Input Shown in Bold
Our TVs come in three models: The 100, 200, and 300. Which do you want? 300 [Enter]
That model has the following features:
Picture-in-a-picture.
Stereo sound.
Remote control.

Program Output with Example Input Shown in Bold
Our TVs come in three models: The 100, 200, and 300. Which do you want? 500 [Enter]
That model has the following features:
You can only choose the 100, 200, or 300.

THE SWITCH STATEMENT
MENU SYSTEM

- `switch` statement is a natural choice for menu-driven program:
  - display the menu
  - then, get the user's menu selection
  - use user input as expression in `switch` statement
  - use menu choices as `expr` in `case` statements
MORE ABOUT BLOCKS AND SCOPE

• Scope of a variable is the block in which it is defined, from the point of definition to the end of the block
• Usually defined at beginning of function
• May be defined close to first use

INNER BLOCK VARIABLE DEFINITION
IN PROGRAM

```cpp
if (income >= MIN_INCOME) {
    // Get the number of years at the current job.
    cout << "How many years have you worked at " << "your current job? ";
    int years;  // Variable definition
    cin >> years;
    if (years > MIN_YEARS)
        cout << "You qualify.\n";
    else
        {  
            cout << "You must have been employed for\n" << "more than " << MIN_YEARS << " years to qualify.\n";
        }
} 
```
VARIABLES WITH THE SAME NAME

• Variables defined inside { } have local or block scope
• When inside a block within another block, can define variables with the same name as in the outer block.
  • When in inner block, outer definition is not available
  • Not a good idea

TWO VARIABLES WITH THE SAME NAME IN PROGRAM

Program 4.30

```c++
1 // This program uses two variables with the same name.
2 #include <iostream>
3 using namespace std;
4
5 int main()
6 {
7    // Define a variable named number.
8    int number;
9    cout << "Enter a number greater than 0: ";
10   cin >> number;
11   if (number > 0)
12     {
13       int number; // Another variable named number.
14       cout << "Now enter another number: ";
15       cin >> number;
16       cout << "The second number you entered was 
17               " << number << endl;
18     }
19    cout << "Your first number was " << number << endl;
20    return 0;
21 }
```

Program Output with Example Input Shown in Bold
Enter a number greater than 0: 2 [Enter]
Now enter another number: 7 [Enter]
The second number you entered was 7
Your first number was 2