Exam 1

- Exam 1 next Thursday.
  - In-class
  - Paper
  - Program
  - Content: Chapter 1 – Chapter 4.

4.1 Relational Operators

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 Expressions

- 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`

Relational Expressions

- Can be assigned to a variable:
  - `result = x <= y;`
- Assigns 0 for false, 1 for true
- Do not confuse `=` and `==`
4.2

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 if Statement

- General Format:

```c
if (expression) statement;
```

The if Statement - What Happens

To evaluate:

```c
if (expression) statement;
```

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

if Statement in Program 4-2

Program 4-2

```c
1 // This program averages three test scores
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main()
7 {
8   int score1, score2, score3; // To hold three test scores
9   double average; // To hold the average score
10   // Get three test scores
11   cout << "Enter three test scores: ">
12   cin >> score1 >> score2 >> score3;
13   average = (score1 + score2 + score3) / 3.0;
14   cout << "The average is: " << average << endl;
15   // If the average is greater than 90, congratulate the user.
16   if (average > 90)
17     cout << "Congratulations! That's a high score!";
18   else
19     cout << "Average is " << average << endl;
20   cout << "The average is: " << average << endl;
21   return 0;
22 }
```

if Statement in Program 4-2

Program Output

- Program output with example input: Shows score in bold.
  - When 3 test scores and will average score: 80 90 70 [Enter]
  - Program output with other example input: Shows in bold.
    - When 3 test scores and will average score: 100 100 100 [Enter]
    - Your average is 100.0
    - Congratulations! That's a high score!
Flowchart for Program 4-2 Lines 21 and 22

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
- 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.
The if/else statement

**General Format:**

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

if/else-What Happens

To evaluate:

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

- If the `expression` is true, then `statement1` is executed and `statement2` is skipped.
- If the `expression` is false, `statement1` is skipped and `statement2` is executed.

The if/else statement and Modulus Operator in Program 4-8

Program 4-8

```c
// This program uses the modulus operator to determine
// if a number is odd or even. In the number is evenly divisible
// by 2, it is an even number. A remainder indicates it is odd.
#include <iostream>
using namespace std;

int main()
{
    int number;
    cout << "Enter an integer and I will tell you if it\'s\n    even or odd.\n    " << endl;
    cin >> number;
    if (number % 2 == 0)
        cout << number << " is even.\n    
    else
        cout << number << " is odd.\n    
    return 0;
}
```

Testing the Divisor in Program 4-9

Program 4-9

```c
// This program asks the user for two numbers, num1 and num2.
// num1 is divided by num2 and the result is displayed.
// For the value 0, if it contains 0, the division does not
// take place.
int main()
{
    double num1, num2, quotient;
    cin >> num1 >> num2;
    if (num2 == 0) // Extra: checking for division by zero
    {
        cout << "Error: Divisor cannot be zero.\n    
    num2 = \n    
    return 0;
```

4.5

Nested if Statements
Nested if Statements

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

Nested if Statements

- From Program 4-10

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

Nested if Statements

- Another example, from Program 4-1

```cpp
20 // Determine the user's loan qualifications.
21 if (employed == 'Y')
22 {
23   if (recentGrad == 'Y') // Nested if
24     cout << "You qualify for the special ";
25     cout << "interest rate.
26   
27 }
28 else // Not a recent grad, but employed
29     cout << "You must have graduated from ";
30     cout << "college in the past twelve";
31     cout << "years to qualify.
32   
33 }
34 
35 else // Not employed
36     cout << "You must be employed to qualify.
37   
38 }
39 
```

Use Proper Indentation!

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"
if/else if Format

```c++
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 in Program 4-13

```c++
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
31   cout << "Your grade is F.\n";
```

Using a Trailing else to Catch Errors in Program 4-14

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

```c++
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
Logical Operators

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

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;&amp;</td>
<td>AND</td>
<td>New relational expression is true if both expressions are true</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>NOT</td>
<td>Reverses the value of an expression – true expression becomes false, and false becomes true</td>
</tr>
</tbody>
</table>

Logical Operators-Examples

```cpp
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
```

The logical && operator in Program 4-15

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

The logical || Operator in Program 4-16

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

The logical ! Operator in Program 4-17

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

Logical Operator-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)
4.9

Checking Numeric Ranges with Logical Operators

- 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!
  ```

4.10

Menus

- **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

Menu-Driven Program Organization

- 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

4.11

Validating User Input
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

---

C++

4.12

Comparing Characters and Strings

Comparing Characters

- 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'

Relational Operators Compare Characters in Program 4-20

Comparing string Objects

- Like characters, strings are compared using their ASCII values

```cpp
10 // Get a character from the user.
11 cin >> processChar();
12
13 // Determine what the user entered.
14 if (processChar() == 'A' && processChar() == 'Z')
15    cout << 'You entered a digit.
16 else if (processChar() == 'A')
17    cout << 'You entered an uppercase letter.';
18 else if (processChar() == 'a')
19    cout << 'You entered a lowercase letter.';
20 else
21    cout << 'That is not a digit or a letter.';
```

```cpp
string name1 = "Mary";
string name2 = "Mark";

name1 > name2 // true
name1 <= name2 // false
name1 != name2 // true
name1 < "Mary Jane" // true
```
4.13 The Conditional Operator

Can use to create short if/else statements

Format: expr ? expr : expr;

First Expression: Expression to be tested

2nd Expression: Executes if first expression is true

3rd Expression: Executes if the first expression is false

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 in Program 4-22

// This program calculates a restaurant’s charges at 5%  
// tax rate. For a minimum of 5 hours, the $5/hour  
// charge applies. Enter time worked.
#include <iostream>
#include <iomanip>

int main()
{
    double PST_RATE = 5.0;  // 5% sales tax
    double HRS_MINIMUM = 5.0;  // Minimum charge
    double HRS_WORKED;
    double charge;  
    int menu;

    std::cout << PST_RATE << std::endl;  // Output sales tax rate
    std::cout << HRS_MINIMUM << std::endl;  // Output minimum charge
    std::cout << std::endl;  // New line

    // Get number of hours worked
    std::cout << "Enter number of hours worked: ";

    // Check number of hours worked
    while (menu > HRS_MINIMUM)
    {
        // Display message
        std::cout << "Enter number of hours worked: ";
        // Get number of hours worked
        std::cout << "Enter number of hours worked: ";
    }

    // Calculate total charge
    charge = HRS_WORKED * HRS_MINIMUM;  // Calculate charge
    // Calculate total charge
    charge = HRS_WORKED * HRS_MINIMUM;  // Calculate charge
    // Display the charges
    std::cout << "The charge is 
"; charge << std::endl;}
The **switch** Statement

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

**switch** Statement Format

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

The **switch** Statement in Program 4-23

Program Input with Example Input Shown in Bold
```
A
```
Program Output
```
The value A is in the second A
```

Program Input with Example Input Shown in Bold
```
B
```
Program Output
```
The value B is in the second A
```

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

**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`

**break** Statement

- Used to exit a `switch` statement
- If it is left out, the program “falls through” the remaining statements in the `switch` statement
break and default statements in Program 4-25

Using switch in Menu Systems

• 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 4-29
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 4-30

```
Program 4-30
1. This program has two variables with the same name.
2. Enter two numbers.
3. Use a single statement.
4. Set first number to input.
5. Set second number to input.
6. while number1 > number2 do
7.     read number1
8.     read number2
9. end while
10. if number1 = number2 then
11.     print "The second number received was "
12.     print number1
13. else
14.     print "The second number you received was "
15.     print number2
16. end if
17. end
```

Program Output with Example Input: (shown in bold).

Enter a number greater than 5 [Input]:
The second number you received was 7
Enter another number: 7 [Input]:
Two first number same...