# CSCI 1380

FEBRUARY 27, 2017

# WEDNESDAY RECAP.

# SEQUENTIAL EXECUTION

```
int main()                                  // execution starts
{
    string name;
    cout << “Enter your name:”;
    cin >> name;
    cout << “Hello “ << name;
}
```

# SEQUENTIAL EXECUTION

```
int main()
{                                          // entering body of the main function
    string name;
    cout << “Enter your name:”;
    cin >> name;
    cout << “Hello “ << name;
}
```

# SEQUENTIAL EXECUTION

```
int main()
{
    string name;                          // reserving space in RAM for name
    cout << “Enter your name:”;
    cin >> name;
    cout << “Hello “ << name;
}
```

# SEQUENTIAL EXECUTION

```
int main()
{
    string name;
    cout << “Enter your name:”;          // prompts the user for input
    cin >> name;
    cout << “Hello “ << name;
}
```

# SEQUENTIAL EXECUTION

```
int main()
{
    string name;
    cout << “Enter your name:”;
    cin >> name;                    // assigns name the inputted string
    cout << “Hello “ << name;
}
```

# SEQUENTIAL EXECUTION

```
int main()
{
    string name;
    cout << “Enter your name:”;
    cin >> name;
    cout << “Hello “ << name;          // prints out Hello Professor
}
```

# SEQUENTIAL EXECUTION

```
int main()
{
    string name;
    cout << “Enter your name:”;
    cin >> name;
    cout << “Hello “ << name;
}
```

# CONDITIONAL EXECUTION: `if`

- Program executes a statement (or set of statements) *if* a certain condition (logical expression) is met.

# EXAMPLE

- "If your final score is greater than or equal to 90, then you get an A."

condition

statement that will execute
(if condition is met)

# EXAMPLE

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXAMPLE

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
            grade = ‘A’;
    }
}
```

if STATEMENT

# EXAMPLE

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
          grade = ‘A’;
    }
}
```

LOGICAL EXPRESSION

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()                                  // execution starts
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{                                          // entering body of the function
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;                    // reserving space in RAM for score
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;                    // reserving space in RAM for grade
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;          // prints “Enter your score:”
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;                          // say user inputs “95”;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)                    // condition is met
    {
            grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
            grade = ‘A’;
    }
}
```

```
// computer enters conditional statement
// and assigns ‘A’ to grade
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
           grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;                          // say user inputs “82”;

    if (score >= 90)
    {
            grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)                    // condition is not met
    {
            grade = ‘A’;
    }
}
```

# EXECUTION

- “If your final score is greater than or equal to 90, then you get an A.”

```
int main()
{
    float score;
    char grade;
    cout << “Enter your score:”;
    cin >> score;

    if (score >= 90)
    {
            grade = ‘A’;
    }
}
```

# EXERCISE

- With a partner, write a program that:
  - asks the user for a number
  - and prints out whether it is positive, negative, or equal to zero

# EXERCISE: DETERMINING MAX

- With a partner, write a program that:
  - compares two numbers of type `int`: `num1` and `num2`
  - and prints out the larger of the two numbers

# LOGICAL EXPRESSIONS

# LOGICAL EXPRESSIONS

- Arithmetic expressions:
  - built with arithmetic operators
  - evaluate to numbers (integers or floating-point)
  - E.g., `3 + 5`

- **Logical expressions**:
  - built with **relational operators**
  - evaluate to `true` or `false`
  - E.g., `3 <= 5`

# RELATIONAL OPERATORS

- Relational operators in C++:
  == (equal to)
  != (not equal to)
  < (less than)
  <= (less than or equal to)
  > (greater than)
  >= (greater than or equal to)

# COMPARING NUMBERS

- Integer and floating-point types can be compared:
  - `8 < 15` evaluates to `true`
  - `6 != 6` evaluates to `false`
  - `2.5 > 5.8` evaluates to `false`
  - `5.9 <= 7` evaluates to `true`

# COMPLEX LOGICAL EXPRESSIONS

# CHECKING MULTIPLE CONDITIONS

- What is the appropriate logical expression for each of the following expressions?
  - free shipping for orders over $25
  - 10 items or less
  - children ages 3 to 11 allowed on play equipment

# CHECKING MULTIPLE CONDITIONS

- What is the appropriate logical expression for each of the following expressions?
  - free shipping for orders over $25
    ```
    orderTotal > 25.0
    ```
  - 10 items or less
    ```
    itemCount <= 10
    ```
  - children ages 3 to 11 allowed on play equipment
    ```
    age >= 3 && age <= 11
    ```

# COMPLEX LOGICAL EXPRESSIONS

- Logical (Boolean) operators enable you to combine logical expressions.
  - not: !
  - and: &&
  - or: | |

# **AND** OPERATION

- "Only children ages 3 to 11 are allowed to play on the equipment."
- `if (age >= 3 && age <= 11)`

| EXPRESSION 1 | EXPRESSION 2 | EXPRESSION 1 && EXPRESSION 2 |
|---|---|---|
| true | true | **true** |
| true | false | false |
| false | true | false |
| false | false | false |

# **OR** OPERATION

- "Only children of age 5+ or 3+ feet tall are allowed to play on the equipment."
- `if (age >= 5 || height >= 3)`

| EXPRESSION 1 | EXPRESSION 2 | EXPRESSION 1 && EXPRESSION 2 |
|---|---|---|
| true | true | **true** |
| true | false | **true** |
| false | true | **true** |
| false | false | false |

# **NOT** OPERATION

- "Only children, who are **not** under 3 feet tall, are allowed to play on the equipment."
- `if (!(height < 3))`

| EXPRESSION | !(EXPRESSION) |
|---|---|
| true | false |
| false | **true** |

# ORDER OF PRECEDENCE

- Relational and logical operators are evaluated from left to right.
- Parentheses can override precedence.

| OPERATORS | PRECEDENCE |
|---|---|
| ! | first |
| *, /, % | second |
| +, – | third |
| <, <=, >=, > | fourth |
| ==, != | fifth |
| && | sixth |
| \|\| | seventh |
| = (assignment) | last |

# NOTE: ASSIGNMENT VS. EQUIVALENCE

- Assignment: `=`
  - For example: `int x = 5;`

- Equivalence: `==`
  - For example: `if (x == 5)`

# EXERCISE: TESTING THE RANGE

- With a partner, write a program that:
  - prompts the user for three integer numbers (call them x, y, and z)
  - and prints out whether x's value is between that of y's and z's

# EXERCISE: TESTING THE RANGE

- Individually, write the program's output given user input of:
  - 1, 2, 3
  - 10, 5, 20
  - 10, 20, 5
  - 3, 2, 1
  - 1, 1, 2