# C++ Syntax & Variables

CS 150 – C++ Programming I
Lecture 2

## Input, Processing and Output

- What do computers actually do?
  - They take input (raw data) and convert (process it), output as information
- Exercise: write a C++ IPO program
- Review: how do we solve this problem?
  - Step 1 – Design (inputs, outputs, calculations)
    - Add IDs (2) and comments to your file
  - Step 2 – Mock up the I/O, use literals for each
    - Run and compare to the handout
- Now, let's talk about variables and values

## Variables as "Memory Boxes"

- Programs store data in variables
- A variable is like a box with a label on it
  - The box is memory, the label is a name
- You can:
  - put something into a box
  - take something out of a box
  - change what is in the box
- Size of the box restricts what you can store

## Values and Data Types

- The data stored inside a variable is called its value
  - A variable has different values as a program runs
- Different kinds of values we can put in a variable
- Different kinds of containers for different kinds of values
- Different sizes of containers for similar kinds of values

## Variables and Values

- C++ values are quantities of data
  - `1`, `3.1459`, `"Steve"`, `'D'`, `true`
- Values represent many kinds of things
  - Integers, real, strings, characters, Boolean, streams
  - The "kind of value" is called a value's type
  - Definintion: A set of bits interpreted according to its type
- Variable: a named storage location that holds a value
  - What names are legal in C++?
  - Aka "objects" in C/C++ (distinct from OO objects)
  - Each variable holds values of one type

## Declaring and Defining a Variable

- To create (declare and define) a variable we give it a name and specify the type of thing it holds
  - `Type name;`
- We can also give the variable an initial value
  - `Type name = value;` *OR*
  - `Type name(value);` *OR*
  - `Type name{value};`
- The equal sign doesn't mean equality as in algebra
  - Copies "stuff" on the right into the variable on the left

## Getting Input

- Let's create a variable to hold the current salary
- What type should it be?
  - C++ has two families of numbers integers and reals
  - Integers are whole (discrete) numbers
  - Real (floating-point) numbers may have a fractional part
- What value should we initialize current salary with?
  - Nothing; we want to get the value from input
  - To get numeric input use the `cin` object:
  - `cin >> myVariable;`

## Processing

- Processing: turning input into desired output.
  - Output: annual & monthly salary, retroactive pay
- 1. Create variables for each output value
- 2. Initialize on line that creates it

```
int var = calculation;      // this, not
int var;                    // this
var = calculation
```

- 3. Algorithms
  - A. Let `annual` be `original` x 1 + `raise percent`
  - B. Let `monthly` be `annual` / `months per year`
  - C. Let `retroactive` pay be `annual` - `original` / `2`

# Magic Numbers, const & Formatting

- We assume that the pay increase is `7.6%`
- Should use named constants not literals
  - "Magic Number" rule: no literals other than `0`, `1`, or `-1`
- Use the `const` qualifier

```
const double kStateRate = .0075;
const double LOCAL_RATE = .000052;
```

- Format real numbers with IO manipulators:
  - `#include <iomanip>`

```
cout << fixed << setprecision(2);
```

- Appears once; modify precision on demand

## Five Variable Concepts

- Declaration: associate name with a type
  - `extern int x; // type of x is int`
  - Variables must be declared to compile
  - See variable `ASSIGNMENT` in homework
- Definition: reserve space for an object or code
  - `int y; // a defining declaration (both)`
- Initialization: provide initial value for object
- Assignment: copy new value into object
- Input: a special form of assignment

## Initializing Variables

- Variable definition doesn't create a value
- You must initialize or otherwise populate
  - Not a syntax error if you forget; logic error
- Three ways to initialize a variable

  -
    ```
    double c{7.3};        // uniform (C++11)
    double a = 2.5;       // NOT assignment
    double b(2e3);        // direct
    ```
- Assignment: `num = 3.159; n2 = {3.159};`
- Input: `cin >> num;`

## Assignment, L-Value and R-Value

- Stores a value in a variable
  - The assignment operator is the = token
  - An expression, not a statement (has a value)
- Assignment statement has 3 parts:
  - Object where the value is to be stored
    - Appears on the left, called an *lvalue* (*el-value*)
    - A region where values can be stored (addressable)
  - The assignment operator ( = )
  - Value to be stored, (*rvalue* - right-value)
- Constants and arrays are non-modifiable lvalues

## Input: Reading a number

- For numeric input use `cin` and extraction operator

```
cin >> variable;
```

  - Read and discard leading whitespace characters
  - Match characters to the type of the variable
  - Stop reading when a mis-match occurs
  - Convert characters to binary data and store in variable
- What if input doesn't match the type?
  - `cin` goes into a fail state. No runtime exception, like Java
- What if there is no input? Program blocks (keyboard)