CSE 2010:
Week 2

Chapter 2: Numerical Data Types & Variables to Represent Them

What this lecture covers:

- Data types
  - What they are.
  - The data types used to represent numerical values.
- Variables
  - What they are
  - Declaring and Initializing them
  - Using them in your programs
  - Assigning them new values

Data Types:
Background and Definition

- **Background:**

  When we are writing a program, we have to represent different types of values depending on the task at hand. C++ is able to represent simple numerical and character values using specific, fundamental data types.

- **Definition:**

  Values in programming have a specific data type that determines the size it will take up in memory, and the type of value that can be stored at that memory location.

Data Types:
Integer and Floating-Point

- **Numeric data:**
  - Integers and floating point aka decimal numbers.
- Integer values:
  - Whole #'s with no fractional part (negatives, zero, positives).

C++ Keyword	Size of type (memory allocated)	Range of values that can be stored in this type
`int`	4 bytes	-2,147,483,648 to 2,147,483,647

- Floating-Point values:
  - Numbers with decimal points in them. "Floating-point" refers to the decimal floating around and changing the value of the number.

C++ Keyword	Size of type (memory allocated)	Range of values that can be stored in this type
`float`	4 bytes	$\pm 1.18 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$
`double`	8 bytes	$\pm 2.23 \times 10^{-308}$ to $\pm 1.80 \times 10^{308}$

Number Types:

Table 1 Number Types

Type	Typical Range	Typical Size
`int`	–2,147,483,648 . . . 2,147,483,647 (about 2 billion)	4 bytes
`unsigned`	0 . . . 4,294,967,295	4 bytes
`short`	–32,768 . . . 32,767	2 bytes
`unsigned short`	0 . . . 65,535	2 bytes
`long long`	–9,223,372,036,854,775,808 . . . 9,223,372,036,854,775,807	8 bytes
`double`	The double-precision floating-point type, with a range of about $\pm 10^{308}$ and about 15 significant decimal digits	8 bytes
`float`	The single-precision floating-point type, with a range of about $\pm 10^{38}$ and about 7 significant decimal digits	4 bytes

(Big C++, 2nd Edition, pg. 39, Table 1)

Variables:
Background & Definition

- **Background:**
  - When writing a program to complete a specific task, it is very likely we must save values throughout the program to use them later. We need a way to store these values in memory and use them whenever we need to.
- **Definition:**
  - Variables are named storage that can be used throughout a program. They allow us to store values in memory and access them later without having to note the physical address in memory.

Variables:

C++ Syntax for variable declaration/definition

- `datatype variableName;`
  - Declares a single variable with no initial value
  - Example:
    ```
    int x1;
    ```
- `datatype variableName, variableName, variableName…;`
  - Declares multiple variables of the same datatype, all with no initial values. Separate each variable name with a comma.
  - Example:
    ```
    int x, y, z;
    ```
- `datatype variableName = initial value;`
  - Defines a single variable with an initial value
  - Example:
    ```
    int age = 30;
    ```
- `datatype variableName = initial value, variableName = initial value…;`
  - Defines multiple variables of the same datatype, each with their own initial values
  - Example:
    ```
    double price1 = 30.99, price2 = 41.55, price3 = 25.25;
    ```
- Notes:
  - You can declare/define as many variables as you want in a single statement, but they all must be the same data type.
  - The “=” character used above is the assignment operator.
    - Format is: variable = value;
      Where it will assign the variable on the left the value on the right.

Variables:
Rules for Variable Names

- Variable names must start with a letter or _, but can then contain a mix of uppercase & lowercase letters, numbers, and _. (No other non-letter characters allowed)
- Cannot be any C++ keywords (`int, float, double, main, class, case`, pg. 960 in textbook)
- Variable names are case sensitive, so the variables:

```
int pennies;
int Pennies;
// would be seen as different variables
```

- Make variable names something significant. It should be clear what the variable represents.

Variables: Integer Values

- C++ keyword : `int`
- When to use `int`
  - Use `int` to represent data that can only be identified in whole numbers
  - age, number of coins, ID numbers, the year,

```
int age; int pennies;
int coyoteID; int year;
```

- Can perform arithmetic with integers (+, -, *, /, %)

```
int x = 30, y = 6, z =7;
int b = x + y; // b = 36
int s = x - y; // s = 24
int t = x * y; // t = 180
int u = x / y; // u = 5
int v = x / z; // v = 4 (Integer Division)
```

- Modulus Operator: %
  - op1 % op2 results in the remainder of dividing op1 by op2
  - Example:

```
int x = 30, y = 6, z =7;
int v = x % y; //v = 0
int w = x % z; // w = 2
```

Variables:

Assigning new values to variables

- Variables are called variables for a reason…their values can change!
- Syntax:

```
variableName = value;
```

  - This value can be any expression that is the same data type as the variable
    - Constant value
    - Arithmetic expression
    - Function call
    - Combination of everything

  Example:

```
int x = 60, y = 10;
x = x + y; // x = 70
x = 100 + y; // x = 110
```

- You can combine the assignment operator with an arithmetic operator to assign a new value

```
int count = 0;
count = count + 1; //count = 1
OR
int count = 0;
count +=1; // count = 1
```

- a op= b ➔ a = a op b
  can be used with -=, *=, /=, %=

Variables:

Incrementing and Decrementing with Unary Operators

- Increment and decrement operators
- ++ increment by 1
- -- decrement by 1
- pre-increment/pre decrement (++x,--x): increment/decrement x by 1, and THEN access value
- post increment/post decrement (x++, x--) : access the value of x, and THEN increment/decrement by 1
- Examples:

```
int x = 6;
int z = x++ * 3;
// After the above two statements, z = 18 and x = 7

int y = 3;
int t = ++y * 5;
// After the above two statements, t = 20, and y = 4
```

Variables:
Floating-Point Values

C++ keyword : `double`

- When to use `double`
  - Use `double` when the value with contain decimal points
  - Monetary ($$) values, physics problems, math problems

```
double speed; double price;
double pi; double volume;
```

- Similarly, to integers, we can use arithmetic operators on these types
(+, -, *, /, %)

```
double length = 9.5;
double width = 10.6;
double area = length * width; //area =
100.7
double x = 5/2; //2.5
```

- Note:
  - Cannot use % with floating point values.
  - Can use fmod(),part of the cmath library

Variables: Constants

- Constants:
  - A constant is a named value that cannot be changed.
  - Use constants in your program when you are representing a fixed value.
- Syntax:

```
const datatype NAME = value;
```

- Example:

```
const double PI = 3.1415926535;
double radius = 0;
cout << "Enter the radius of the circle: ";
cin >> radius;
cout << "The area of the circle is: " << PI * r * r << "\n";
```

Let’s come up with the steps to solve the following problem

Write a C++ program that asks the user the number of pennies, nickels, dimes, and quarters they have. Then, display their total to them.

- Steps to take
- What we need for each step: