Chapter 3
Variables/Assignments

FORDHAM

CISC 5004 - Programming C++
Sam Kamens
Fall, 2022

Variables and Assignments (General)

- People on Bus
- Variables and Assignments
- Variables on both sides of the assignment

Variables (int)

FORDHAM

Variables (int)

- Declaration
  - Tells the compiler the variable name and type
  - e.g. `int userAge;`
  - Good practice to initialize when defining the variable:
    - `Int userAge = 0;`
- Refers to a memory location
  - Note: Compilers can optimize this away, but the concept is worthwhile

Assignments

- Set the value of the variable on the left hand side to the result of evaluating the right hand side
- Expression
  - Integer Literal (no commas allowed!)
  - Formula/Calculation
- PA: Assigning a variable
- Increment/decrement
  - `x++` is equivalent to `x = x + 1`
  - `x--` is equivalent to `x = x - 1`
- Operate and assign
  - `x += 7, x -= 10`

Common Errors

- Read from uninitialized variable
  - ints are usually initialized to 0 by default, but not guaranteed
- Assignment statements in reverse
  - `numKids + numAdults = numPeople` ⇒ Compiler Error
  - `numCats = numDogs` ⇒ Possible Logic Error

Identifiers

FORDHAM

Rules for Identifiers

- Name created by a programmer for a function or variable
- Contains characters from a-z, A-Z, underscore, and 0-9
- Must start with letter or underscore
- Case Sensitive
- Reserved Word
  - aka keyword
  - Part of the language, so not allowed as an identifier
  - C++ Reserved Words/Keywords

Variable Naming

- Follow the coding standards for your team/organization/company
- Local variables always start with lowercase letters
  - Globals may start with uppercase letters
- Two common styles:
  - Camel Case (used in zyBook)
    - Capitalize first letter of second and subsequent word
    - e.g. `numStudents`, `peopleWithRedShoes`
  - Underscore separated
    - e.g. `num_students`, `people_with_red_shoes`
- Good practice
  - Create meaningful variable names
  - Minimize use of abbreviations
  - But also avoid over-long names

Arithmetic Expressions (General)

FORDHAM

Basics

- Expression: Item or combination of items that evaluates to a value
  - e.g. 2 * (x + 1)
- Literal: A specific value in code like 2
- Operator: A symbol that performs a built-in calculation
  - Arithmetic Operators: +, –, *, /

Evaluation

- Expression Evaluates to a value
  - Incidentally, an assignment is an expression
    - `x = 9` actually evaluates to 9
- Precedence rules (PEMDAS)
  1. ( )
  2. unary -
  3. * / %
  4. + -
  5. left-to-right
- Good practice: Use parentheses to explicitly specify order of evaluation
  - `y = (m * x) + b`
  - `x + y + z`

Arithmetic Expressions (int)

- Expressions Example
- Style:
  - Single space around operators
  - `x = x + 2` (instead of `x=x+2`)
- Compound Operators
  - `+=, -=, *=, /=, %=`

Floating Point Numbers (double)

FORDHAM

Floating-point Numbers

- Real Number (containing a decimal point)
  - Decimal point may appear anywhere (i.e. it “floats”)
- Variable of type `double` holds a floating point number
  - `double averageGrade;`
- Floating-point literal: Number with a fractional part
  - e.g. `1.0`, `0.0`, `99.573`
- Good practice: Don’t omit leading 0
  - `0.5` vs `.5`
- Travel Time Example

Which type should you choose?

- Integers (int)
  - Things that can be counted (2 pizzas, 101 dalmations)
- Floating point (double)
  - Non-integer measurements (98.6 degrees, 26.2 miles)
  - Fractions of countable items (2.1 kids per household)
  - Caveat: using floating point for money is problematic

Division by Zero

- Different behavior between `int` and `double`
- `int`
  - Compile warning
  - Runtime exception
- `double`
  - `inf`, `-inf`, `NaN`

Manipulating output of floating point numbers

- iomanip
  - `#include <iomanip>`
- `cout << fixed << setprecision(3) << 3.1244 << endl;`
  `cout << 2.1 << endl;`
- fixed
  - Fixed floating point (i.e. do not use exponential notation)
  - https://cplusplus.com/reference/ios/
- setprecision
  - How many decimal places to print
  - https://cplusplus.com/reference/iomanip/

Scientific Notation for Floating Point Literals

- e.g.

```
// Approximation of atoms per mole
double avogadrosNumber = 6.02e23;
```

Constant Variables

- Variables whose values can not change after they are initialized
- Useful for numeric constants used in code
- Example: Lightning Distance

Math Functions

- Standard math library (cmath, cstdlib)
- PA: Using a math function
- Common Functions:

Function	Behavior	Example
sqrt(x)	Square root of x	sqrt(9.0) evaluates to 3.0.
pow(x, y)	Power: x^y	pow(6.0, 2.0) evaluates to 36.0.
fabs(x)	Absolute value of x	fabs(-99.5) evaluates to 99.5.

- Full list: http://www.cplusplus.com/reference/clibrary/cmath/
- Function calls in an argument

Integer Division and Modulo

- Integer division does not result in a fraction
  - Floating point result if at least one operand is a floating point type
- Divide By Zero: runtime error (Floating point exception)
  - Compare to floating point result (inf)
- Modulo (%): Remainder
  - Minutes to hours/minutes
  - Random number in range
  - Getting digits
  - Phone number prefix

Type Conversions

- Conversion of one data type to another.
- Automatic conversions are known as implicit conversions
  - int-to-double
  - double-to-int: Fraction is simply dropped
- Assignment of int to double works, but is discouraged
  - Discouraged: `double dVar = 2`
  - Better: `double dVar = 2.0`

Type Casting

- Explicit convert an item's type
  - `static_cast<type>(expression)`
  - Using type casting to force floating point division
- Common errors
  - Not casting arguments
  - Casting result instead of operands

Binary

- Decimal numbers: base 10
- Binary numbers: base 2

Characters

- Type `char`
  - Stores a single character
  - Character literal: ‘c’
- Getting a character from input
- Actually stored as an integer
- ASCII - encoding of characters as numbers
  - http://www.asciitable.com/

Characters

- Escape Sequences

Escape sequence	Char
\n	newline
\t	tab
\'	single quote
\"	double quote
\\	backslash

Common Errors

- Double quotes
- No quotes

Strings

- String: A sequence of characters
- String literal: character sequence in double quotes (`“Hello, World!”`)
- Declaring and assigning a string
- Whitespace:
  - `cin >> stringVariable` skips initial whitespace, but only captures up to the first whitespace character
- getline() function
  - Captures entire line - `getline(cin, stringVar)`
- Combining cin and getline() can be tricky

Integer Overflow

- Standard it (32 bits) can store values up to about 2 billion
- Assigning a larger value will cause overflow
- Example: Store 256 to an 8 bit int:
  - 256 = 100000000 (9 bits)
  - Result will be 0
- May be detected at compile time if using literals
  - Otherwise, be careful of calculations
- Can use `long long` (8 bytes)
  - Believe it or not, `long` is often the same as `int` (4 bytes)

Numeric Data Types

- Integer types
- Floating point types
  - Representation

Unsigned

- Instructs the compiler that no negative values will be stored
- Increases maximum values
  - Unsigned types

Random Numbers

- `rand()`
  - Returns a random number from `0` to `RAND_MAX`
- Use modulo (`%`) to limit the number of possible values
- Specific range
- Important: `rand()` is actually pseudo-random
  - seed: `srand()`
  - Often `srand(time(0))`

Auto

- Compiler determines what the data type of a variable is
- `auto i = 5;`
- `auto j = 12.0;`
- http://en.cppreference.com/w/cpp/language/auto

Style Guidelines

- zyBooks Style
- Google Style Guide

Labs

- Using math functions
- Simple statistics
- Convert to dollars