The Basics:
Variables and data types in C++

William Hendrix

Lecture 2

Outline

- Variables
- Basic data types
- Arithmetic

Before using variables

- Variables must be *declared* before being used
- Syntax:

```
[variable type] [variable name];
```

- `[variable type]` can be any of:
  - `int`, `double`, `char`, `bool`, `long`, `long long`, `short`, `float`, `wchar_t`
  - `unsigned` integral types
  - `string`
  - Other classes
  - Enumerated or user-defined types
  - Pointer to or array of any other types

Today

- `[variable name]` must:
  - Start with a letter or underscore
  - Contain letters, numbers, or underscores
  - *Convention*: variables start with lowercase letters
  - Multiple words: `mixedCase` or `use_underscores`

Using variables

- Assigning values to variables
  - Syntax: `[variable name] = [value or expression];`
    - E.g., `sum = 0;`
    - Can also be combined with declaration: `int sum = 0;`
- Using variables
  - Variable name in code is replaced with its current value
  - E.g., `sum = number;`
  - The variable on the LHS can appear on the RHS
  - Frequent construction: `var = var + 1;`
    - Shortcut: `var++;` or `++var;` or `var--;`
    - Can add values using `var += increment;`
      - Works with most operators `-=, *=, /=,` etc.
  - Anomaly: variable assignments *have a value*
    - `sum = total = 0;`
    - `legal_but = (a_really = 2) + (bad_idea = 3);`

Variable types

- Integral types
  - Positive/negative whole numbers
  - In order of increasing size: `short`, `int`, `long`, `long long`
    - Size (in bytes) is not defined by the language
    - `int` are usually 4 bytes, `short` 2 or 4, `long` 4 or 8, `long long` 8
    - 2 bytes: -32,768 to 32,767
    - 4 bytes: -2,147,483,648 to 2,147,483,647
    - 8 bytes: -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
  - Can also be unsigned (`unsigned int`)
    - Doubles positive range (e.g., 0 to 65,536)

- Boolean types (true/false)
  - `bool`: 1 byte
  - `true`/`false`
    - Represented internally as 1 (`true`) and 0 (`false`)

Non-integral types

- Decimal numbers
  - `double` (8 bytes), `float` (4 bytes)
  - "Floating point": similar to scientific notation, but in binary
- `double`
  - Accurate to ~15 decimal digits
  - Range is approx. $\pm 10^{308}$
  - Can represent values as small as 10^{-308}
- `float`
  - Accurate to ~7 decimal digits
  - Range approx $\pm 10^{38}$ down to 10^{-38}
  - Not commonly used any more
- Roundoff errors are possible with either
  - We'll discuss a potential solution on Monday

Character types

- Letters, digits, symbols
- `char`: 1 byte (always)
  - Characters represented in code with single quotes: `'x'`
  - Encoded using ASCII codes (0-255)
    - E.g., `'0'`–`'9'`: 48-57, `'A'`–`'Z'`: 65-80, `'a'`–`'z'`: 97-122
- Special characters
  - `'\n'`: new line
  - `'\r'`: carriage return (not recommended)
  - `'\t'`: tab
  - `'\\'`: backslash
  - `'\''` and `'\"'`: single and double quotes
  - `'\[return]'`: no character (used to break long lines in a program)
  - `'\0'`: null character; not printed
- `wchar_t` ("wide character type"): 2 bytes
  - Used for non-English letters and symbols

Using C++ like a calculator

- Basic operations
  - Addition (+), subtraction (−), multiplication (*)
    - Combining numbers of the same types produces output of the same type
    - Combining numbers of different types produces output of the more general type
      - bool -> char -> wchar/short -> int -> long -> float -> double
  - Division (/)
    - Type rules are similar to addition
    - Integer types *do not* become floating point
    - Remainders are discarded unless you convert to floating point first
  - Modulus (%)
    - Outputs value of remainder when dividing
    - Only defined for integer types
    - Not as well defined for negative values
- Order of operations: (), */%, then +−

Tonight

- **Recommended reading:** Sections 3.1-3.5