Streams, Locals, and Flow of Control

Kate Gregory
www.gregcons.com/kateblog

Libraries

- **Some languages provide keywords to perform specific tasks**
  - Print, eval, call
- **C++ uses functions and classes for just about everything**
  - Even places that don't look like function calls
- **Both functions and classes can be distributed in a library**
  - Linked into your app along with your object files, or included into your code to be compiled
- **A library called The Standard Library comes with your tools**
  - And the tools know where to find it

Stream I/O

- **Input / Output**
  - Keyboard/screen
  - Files
  - Other sources / targets that support streaming
- **Contrast to "record based" or "fixed" I/O**
  - Database
  - One line in the middle of a file
- **C++ supports stream I/O with operators**
  - >> sends something to a stream
  - << reads something from a stream

Exercise

- **Write your own version of Small that prints out some words and some numbers.**
  - Use double quotes " not single ' around words and groups of words
  - Use as many std::endl as you want
  - The numbers can be calculated (2+2) or just type them (42)
  - Try some non-integer numbers too

- **Don't ask for the impossible**
  - 3/0
  - "hello" + 2
- **Errors for these won't make sense to you yet**

Include

- Including a file into your application

```
#include <iostream>
```

- Opens up a world of libraries and capabilities
  - For libraries you wrote, would also need to link in the library
- Good libraries are in a namespace

```
std::cout << 2+2 << std::endl << "Hello!";
```

- Prevents conflicts if other libraries use the same names

Local variables

- **Variables in C++ have a type**
  - String, number, date, Employee, etc
  - Some types are built into the language
  - Some are User Defined
    - Some “users” are actually library writers
- **Variables must be declared before they are used**
- **Built in types are not initialized for you**
  - User defined might be
- **The compiler enforces a number of rules related to type**

Type Safety

- **C++ enforces type**
  - Variables have type
  - Expressions have type
- **It is ok to "promote"**
  - Put an integer (eg 3) into a float
- **You will be warned if you "demote"**
  - Put a floating point number (eg 4.9) into an integer
- **Some combinations are just not allowed**
  - Put a string into an integer
  - Multiply a string and a float

Flow of Control

- **Normally code is executed from top to bottom**
- **A number of keywords change this**
  - if
  - else
  - while
  - for
- **These keywords work with logical expressions**
  - `(x > 0)`
  - `(y-2 < b)`
- **Operators to compare two operands:**
  - `> >= < <= == !=`
  - Result is `true` or `false`

Exercise

- **Write a "guess my number" game**
  - Hardcode the answer in your code
    - `int answer = 7;`
    - You can change the number, build, and run again
  - Ask the user to enter a guess
  - Let them know if they guessed too high, too low, or got it
  - Keep going until they get it
  - Don't try error checking yet
    - When you test it, be nice

Summary

- The Standard Library provides all you need to read from the keyboard or write to the screen
- In C++ all variables have a type that does not change
- All expressions also have a type
- The compiler enforces a number of rules related to the type system
- Users (you, or a library writer) can define new types
- The keywords if, else, while, and for control the lines that execute in your program
- These small building blocks can build a real application