CPSC 427a: Object-Oriented Programming

Michael J. Fischer

Lecture 2
September 6, 2011

C++ Language Design Goals

Why did C need a ++?

Chapter 2 of Exploring C++

1. C was designed and constructed a long time ago (1971), as a language for writing Unix.
2. The importance of data modeling was very poorly understood at that time.
3. Data types were real, integer, character, and array, of various sizes and precisions.
4. It was important for a C to be powerful and flexible, but not to have clean semantics.
5. Nobody talked about portability and code re-use.

Today, we demand much more from a language.

C++ was Designed for Modeling

Design goals for C++ (Bjarne Stroustrup)

1. Provide classes (replacing structs) as a means to model data.
2. Let a class encapsulate data, so that its implementation is hidden from a client program.
3. Permit a C++ program to link to libraries from other languages, especially FORTRAN.
4. Produce executable code that is as fast as C, unless run-time binding is necessary.
5. Be fully compatible with C, so that C programs could be compiled under a C++ compiler and still work properly.

General properties of C++

- Widely used in the real world.
- Close to the machine and capable of producing efficient code.
- Gives a programmer fine control over the use of resources.
- Supports the object-oriented programming paradigm.
- Supports modularity and component isolation.
- Supports correctness through privacy, modularity, and use of exceptions.
- Supports reusabale code through derivation and templates.

C++ Extends C

- C++ grew out of C.
- Goals were to improve support for modularity, portability, and code reusability.
- Most C programs will compile and run under C++.
- C++ replaces several problematic C constructs with safer versions.
- Although most old C constructs will still work in C++, several should *not* be used in new code where better alternatives exist.

  Example: Use Boolean constants `true` and `false` instead of 1 and 0.

Some Extensions in C++

- Comments // (now in C99)
- Executable declarations (now in C99)
- Type bool (now in C99)
- Enumeration constants are not synonyms for integers
- Reference types
- Definable type conversions and operator extensions
- Functions with multiple methods
- Classes with private parts; class derivation.
- Class templates
- An exception handler.

Tools

Low-level: Command line tools

- A text editor such as `emacs` or `vi`.
- The compiler suite: `g++`.
- Project management: `make`

High-level: Integrated Development Environments (IDEs)

- Integrate various low-level tools
- Facilitate development cycle

Recommended IDE's

The following are all open source and are installed on the Zoo.

Geany	Easy to use. Good for small projects.
CodeBlocks	Good general purpose IDE. Many advanced features, well-engineered, but not well-supported on Mac OS X.
Eclipse/CDT	Powerful, well-supported IDE, somewhat brittle, but getting better all the time.

Example

Generic Insertion Sort

Two implementations of simple insertion sort:

1. **C version:** Written in object-oriented style to the extent possible in C.
2. **C++ version:** Similar code but with C++ support

C version

See code demo `02-InsertionSortC`.

C++ version

See code demo `02-InsertionSortCpp` and following notes.