Introduction to C++

Informatica III – parte A

A. Gargantini

History

- C++ is an object-oriented extension of C
- C was designed by Dennis Ritchie at Bell Labs
  - used to write Unix
  - based on BCPL
- C++ designed by Bjarne Stroustrup at Bell Labs
  - His original interest at Bell was research on simulation
  - Early extensions to C are based primarily on Simula
  - Called “C with classes” in early 1980’s
  - Popularity increased in late 1980’s and early 1990’s
  - Features were added incrementally
- Classes, templates, exceptions, multiple inheritance, type tests...

Design Goals

- Provide object-oriented features in C-based language, without compromising efficiency
  - Backwards compatibility with C
  - Better static type checking
  - Data abstraction
  - Objects and classes
  - Prefer efficiency of compiled code where possible
- Important principle
  - If you do not use a feature, your compiled code should be as efficient as if the language did not include the feature.

What is Data Abstraction?

- Abstract Data Types (ADTs)
  - type implementation & operations
  - hidden implementation
- types are central to problem solving
  - Not procedures like in C
- a weapon against complexity
- built-in and user-defined types are ADTs

How Well are ADTs Supported in C?

- Does **C** enforce the use of the ADTs interface and the hiding of its implementation?
- *No*

C++

- C++ is a superset of C, which has added features to support object-oriented programming

- C++ supports classes
  - things very like ADTs

How successful?

- Given the design goals and constraints,
  - this is a very well-designed language
- Many users -- tremendous popular success
- However, very complicated design
  - Many specific properties with complex behavior
  - Difficult to predict from basic principles
  - Most serious users chose subset of language
    - Full language is complex and unpredictable
  - Many implementation-dependent properties
  - Language for adventure game fans

Further evidence

- Many style guides for using C++ “safely”
- Every group has established some conventions and prohibitions among themselves.
  - don’t inherit implementation
  - SGI compiler group -- no virtual functions
  - Others
-

Overview of C++

- Additions and changes not related to objects
  - type bool
  - pass-by-reference & the Copy-Constructor
  - user-defined overloading
  - function template
  - exception handling
  - …

OO Programming Languages

- Four main concepts:

1. **Abstraction**: implementation details hidden inside a program unit with a specific *interface*. The interface is a set of public functions (or methods) over hidden data.
2. **Inheritance**: reusing the definition of one kind of object to define another kind of object.
3. **Dynamic lookup**: a method is selected at run time, according to the *implementation* of the object, not some static property of the pointer/var used to name the object.
4. **Subtyping** is a relation on types that allows values (or objects) of one type to be used in place of values (or objects) of another.

Inheritance Is Not Subtyping!

"Subtyping is a relation on interfaces, inheritance is a relation on implementations."

C++ Object System

- Object-oriented features

1. Classes and Data Abstraction
2. Encapsulation
3. Inheritance

– Single and multiple inheritance

– Public and private base classes

4. Objects, with dynamic lookup of virtual functions
5. Subtyping

– Tied to inheritance mechanism

– A will be recognized by the compiler as a subtype of B only if B is a public base class of A