High Level Parallel Programming

Introduction to C++

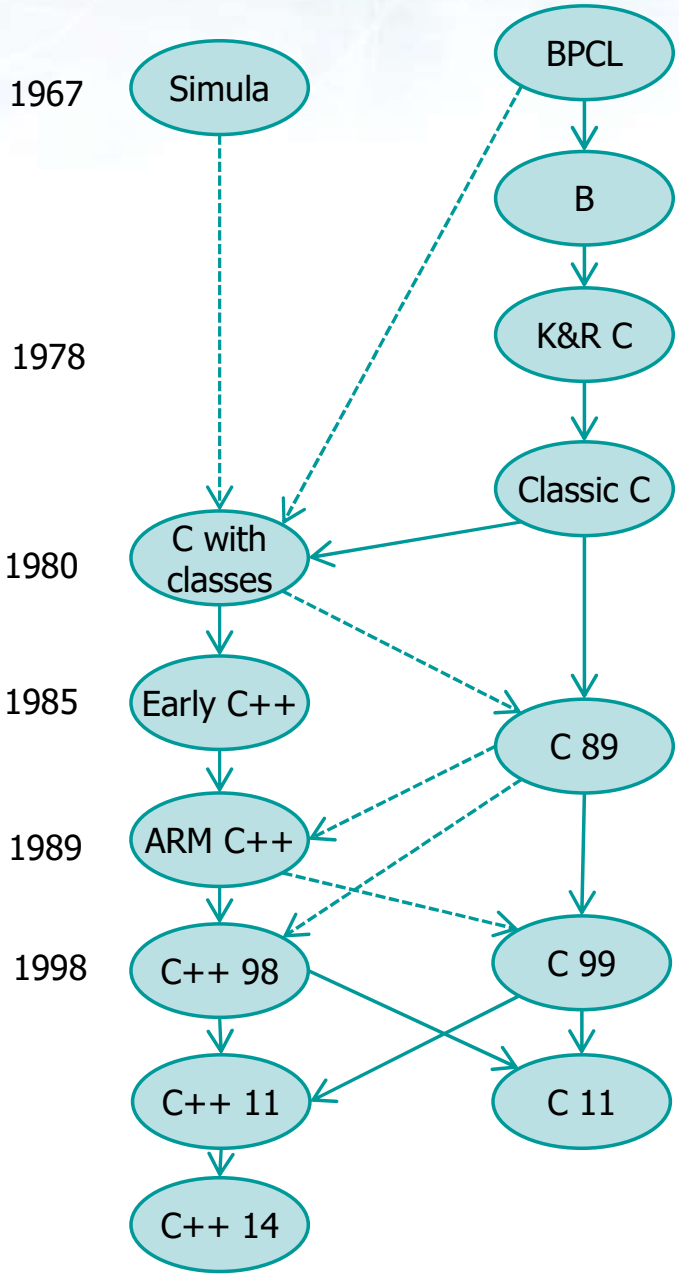
Alessandro Savino and Stefano Quer
Dipartimento di Automatica e Informatica
Politecnico di Torino

History of C++

- C++ standardized versions
  - United States
    - American National Standards Institute (ANSI)
  - Worldwide
    - International Organization for Standardization (ISO)

C++ is not a static language: Several versions exist

From version C++11, C++ includes native multi-threading (without external libraries)



History of C++

- C++ standardized versions
  - Early 1980s
    - Extension of C due to Bjarne Stroustrup (Bell Labs)
    - https://www.youtube.com/watch?v=JBjjnqG0BP8
  - Provides capabilities for object-oriented programming
    - Objects are reusable software components
      - Model items in real world
    - Object-oriented programs
      - Easy to understand, correct and modify
  - Hybrid language
    - C-like style
    - Object-oriented style
    - Both

History of C++

- C++ structure
  - Programs
    - Built from pieces called classes and functions
  - Standard library
    - Rich collections of existing classes and functions
  - "Building block approach" to creating programs
    - "Software reuse"

A C++ program is build as a C program …

… but it has an extremely rich set of libraries (classes, containers, templates, many algorithms, etc.)

Structured Programming

- Structured programming (1960s)
  - Disciplined approach to writing programs
  - Clear, easy to test and debug, and easy to modify
  - Pascal
  - Ada
    - 1970s - early 1980s: US Department of Defense (DoD)
    - Multitasking
      - Programmer can specify many activities to run in parallel

The Key Software Trend: Object Technology

- Objects
  - Reusable software components that model real world items
  - Meaningful software units
    - Date objects, time objects, paycheck objects, invoice objects, audio objects, video objects, file objects, record objects, etc.
    - Any noun can be represented as an object
  - More understandable, better organized and easier to maintain than procedural programming
  - Favor modularity

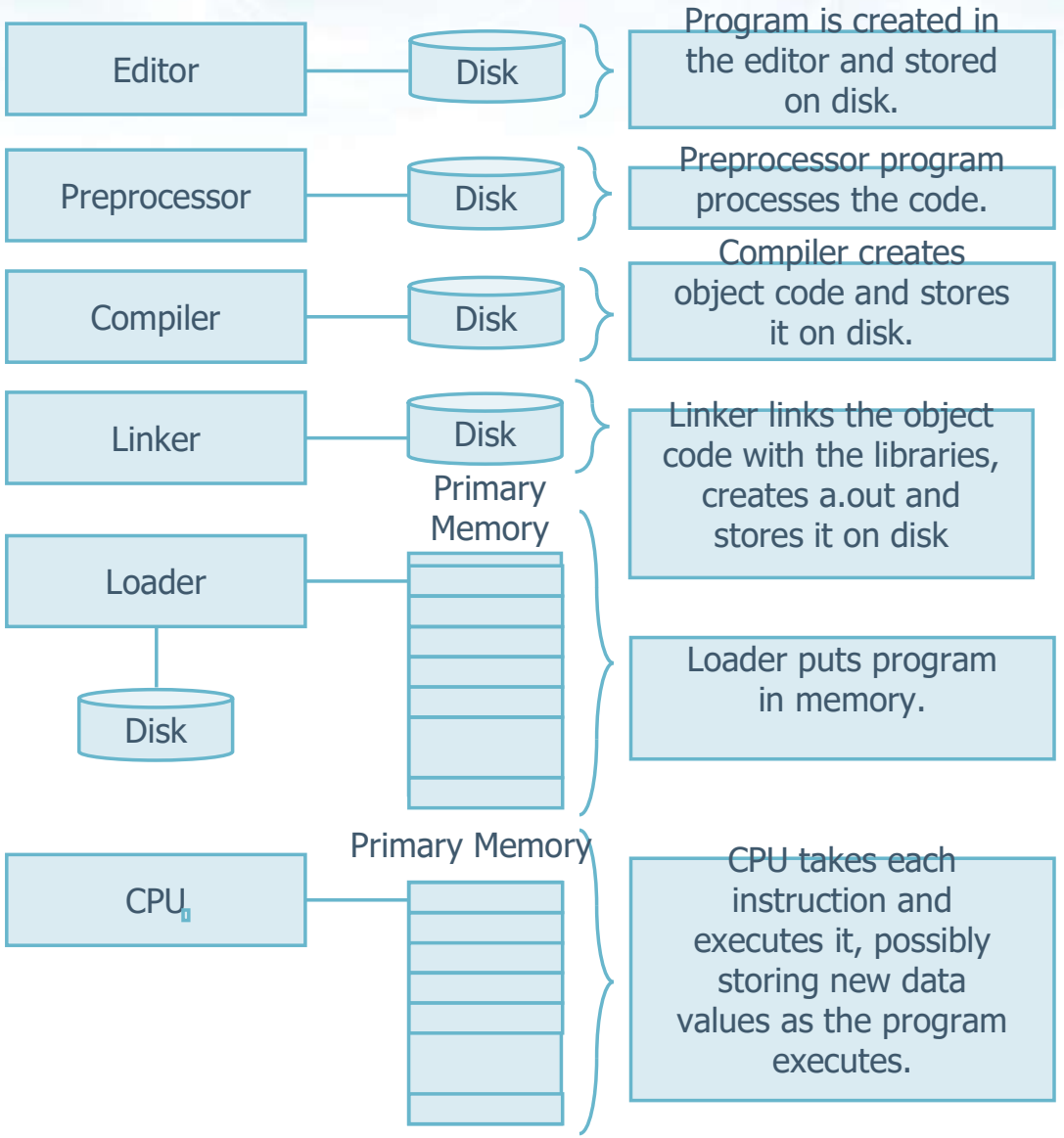
Basics of a Typical C++ Environment

- C++ systems
  - Program-development environment
  - Language
  - C++ Standard Library

Many IDE are available:
Visual Studio, C-Lion, Eclipse, etc.

Basics of a Typical C++ Environment

- Phases of C++ Programs
  1. Edit
  2. Preprocess
  3. Compile
  4. Link
  5. Load
  6. Execute



Please, refer to "Algorithms and programming" for further details

C++ Perspectives

- Object Oriented Programming support
  - Encapsulation
  - Composition
  - Inheritance
  - Polimorphism
- Support for
  - Structured Programming
  - Generic Programming
  - Functional Programming

Recalled notions from the "Object Oriented Programming" course

Programming in a better way with generic data-types (using templates)

How to reduce the amount of code written (important for future C++ developments)

C++ Versions

- Standard has been updated several times
  - Past versions 2011, 2014, 2017
  - New algorithms and data structures
    - The Boost libraries
      - New algorithms, new data-structures, etc.
      - Sooner or later included in C++ directly
  - Lambda programming and other modern languages improvements
  - Last major update in 2020

The core of functional programming

C++ v17

- Main innovations on v11 and later
  - Smart Pointers
    - More expensive yet safe (dynamic) memory management
  - Containers
    - Dynamic containers, not only vectors
  - Exception management
    - Deal with all possible causes of error
  - Lambda function
    - To compact the code as in modern *script languages
  - Modern multiplatform programming and synchronizing
    - Portable on different OS