C-based Languages

- ***C++*** includes all the features of C, but adds classes and other features to support object-oriented programming
  - 1983: C-with-classes redesigned into C++
  - 1985: C++ compilers made available
  - 1989: ANSI/ISO C++ standardization starts
  - 1999: ANSI/ISO C++ standard approved
  - Hence the c99 compiler you will use later

Why C++ and OOP

- Note that, ideally, a sorting method (i.e., a "way" of doing sorting) should work *regardless* of whatever is being sorted.
  - Words; Numbers; Cards; Dates; Times;
  - Think about your experiences on Amazon, eBay, Expedia about the sorting preferences (e.g., prices, auction/connection time)

The need for abstraction

Abstraction involves determining the “least-common denominator” held in common by sets of data/object or functionality/method within a program.

- can be re-used

- can be shared

- can be extended

Strengths of C/C++

Portable: usually does not requires a significant change when the C/C++ programs are ported to different machines (ranging from super-computer to embedded systems)

- C compilers are small and easy to be included in any application development environment

Strengths of C/C++

- **Powerful**: has a large collections of data types/classes and operators/methods.
- **Flexible**: C imposes very few restrictions on the use of its features
- **Standard Library**: almost universal

Strengths of C/C++

Integrated well with all existing platforms: including Microsoft C/C++, and popular UNIX variant (known as Linux and Android) and MacOS (and hence the iOS)

Programming is never easy

- Some mistakes may not be detected by the compilers
- “No errors, No warnings” from compilers is just half-way done to finish the programming
- Run-time errors are hard to debug
- Infinite loop and program crash are possible outcomes

Effective Use of C/C++

- Use software tools (debuggers, etc.) to make programs more reliable.
- Take advantages of existing code libraries.
- Adopt a sensible set of coding conventions.
- Avoid “tricks” and overly-complex codes.
- Stick to the standards (to maximize the portability).

Software Engineering

The term *software engineering* refers to the study of software development on large scales.

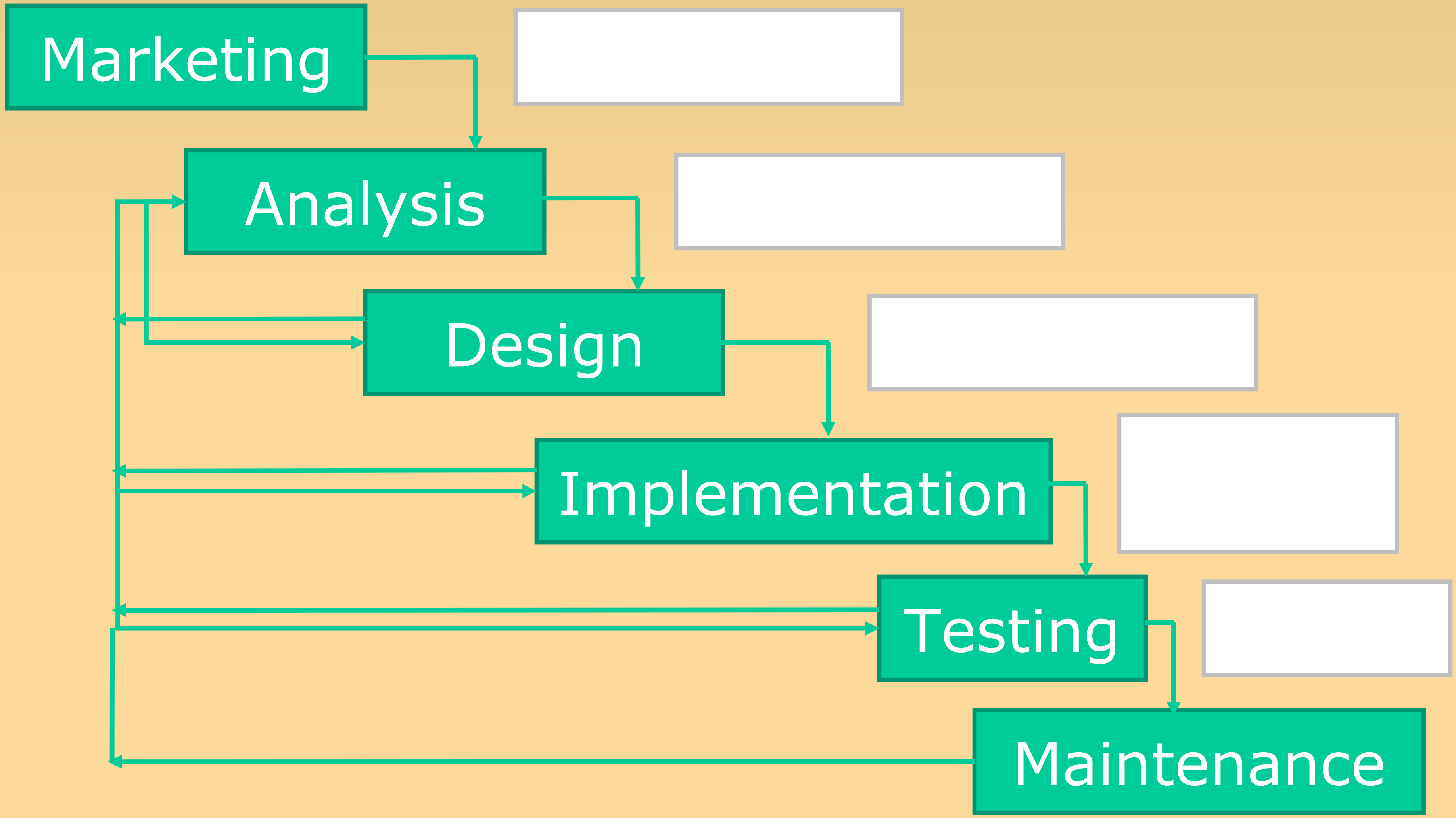
Few programs these days are written by lone, individual programmers.

- Instead, programs are often written by large teams who must coordinate their efforts.

Software Engineering

- The way you approach *program design* matters.
  - Programs are quite often large, complex structures.
    - Many modern video games have teams of over **100** people working on them for over a year.
    - If the program is being written from scratch, make that ***five*** years.

The Waterfall Model



Software Engineering

- The way you approach *program design* matters.
  - For such large projects, it is important to plan your program's structure largely in advance.
    - It is very common for one programmer to use code that another one wrote, sight unseen.
    - Any team member must know what will be available to each component, as well as what form any important data will be in.

Software Engineering

The way you approach *program design* matters.

- It is important to constantly test your program components.
  - The longer you wait to find errors – and you ***will*** have errors – the harder it is to trace their source.
  - Time is money – the more time you waste on errors, the less time left for adding the features that will make your product shine.

Testing: Why?

- The first flight of the ESA Ariane 5: $1 billion in damages.
  - At its core, the error was triggered by an arithmetic overflow exception… which was the result of improper code reuse.
  - The rocket self-destructed a mere 37 seconds after launch.

Testing: Why?

Other known software catastrophes:

- 1999 - Mars Climate Orbiter - wrong units
- 1999 - Mars Polar Lander – engine cutoff too early, SW bug
- 2004 - Mars Rover – too many files opened
- 2006 - Mars Global Surveyor – battery failure due to SW error

Necessary Skills

- Prior programming experience
  - Not necessarily in Java or C/C++, but it helps greatly.
- Willingness to work in groups
- Ability to tolerate frustration
  - Rule #1 of programming: You will make errors.
- Ability to see both the big picture and local, detailed specifics.

What we hope

Eventually, you will have the ability to …

(1) see things abstractly and apply OOP

(2) balance both performance and functional requirements of the software systems

(3) continue the interest on computer science and join the software industry !!

- 77 of world's top 100 software companies are head-quartered in USA (IDC Data 2009) …