Chapter 1

Language Levels

Where do they all go?

- High-level
  - Close to problem
  - System independent
  - Java, C#
  - FORTRAN, COBOL, C++

- Low-level
  - Close to system
  - Doesn’t reflect problem
  - Assembler
  - Machine

Relationship Between C and C++

C++ is (almost) a perfect subset of C

- Improved language features: prototypes, const, inline, references, new, delete
- Main functions, syntax, operators, flow of control, organization, compilation, libraries
- Object-oriented model
- Classes, encapsulation, inheritance, polymorphism, RTTI, templates, ctors/ctors, overloading

Advantages

- Smaller generated code
- More portable (machine independent)
- Better diagnostics (debugging)
- Can implement "very high-level" languages
- 10 to 100 times slower than a compiled program!!
- Hybrid systems are 3 to 10 slower

Interpreter Operation

Dynamic translation

- CPU executes the interpreter
- Program is data to interpreter
- Parses each statement in the program (each every time it is executed)
- Carries out the program statements
- Advantages
  - Smaller generated code
  - More portable (machine independent)
  - Better diagnostics (debugging)
  - Can implement "very high-level" languages
  - 10 to 100 times slower than a compiled program!!
  - Hybrid systems are 3 to 10 slower

Compiling Programs

Multi-file programs (compare with Fig. 1.10, p. 15)

- C was created by Dennis Ritchie in the early 1970s to write a portable version of Unix
- C and Unix came together in 1973 on a DEC PDP-11/20 running the 5th edition of Unix
- The line between C and Unix is not distinct (although they are governed by two standards: ANSI and POSIX)
  - ANSI standard
    - Committee was established in 1983
    - Standard adopted 1989 and became available in 1990
    - Is the base document for the ANSI C++ standard
    - Many features of C++ were back-ported to C either because they were good features or to avoid gratuitous differences

The Origins of C

A brief history

- C was created by Dennis Ritchie in the early 1970s to write a portable version of Unix
- C and Unix came together in 1973 on a DEC PDP-11/20 running the 5th edition of Unix
- The line between C and Unix is not distinct (although they are governed by two standards: ANSI and POSIX)
  - ANSI standard
    - Committee was established in 1983
    - Standard adopted 1989 and became available in 1990
    - Is the base document for the ANSI C++ standard
    - Many features of C++ were back-ported to C either because they were good features or to avoid gratuitous differences
### C Is A Minimal Language

- Lean and mean
  - 32 keywords (27 from K&R and 5 added by ANSI)
  - Does not provide OS/kernel services
    - Networking
    - Multiprocessing/Multithreading
    - Interprocess communication or synchronization
  - Provides basic programming features
    - Variables, expressions, statements, control structures, and functions
    - Minimal compile-time and little run-time error checking
    - Typed but not strongly so—strict type compatibility is not enforced
  - Advanced programming features, including I/O, are provided by libraries (also governed by the ANSI standard)

### Algorithms I

#### A recipe for Fishin’ Cookies

- **Ingredients**
  - ½ C shortening
  - 1 C Sugar
  - 2 well-beaten eggs
  - 1 tsp vanilla
  - 1/4 C milk
  - 3 C flour
  - 2 tsp baking powder
  - 1/4 tsp salt

- **Steps**
  - Cream together sugar & shortening
  - Add eggs
  - Add vanilla & milk and mix
  - Sift together flour, baking powder, & salt; add to liquid and mix
  - Roll dough and cut
  - Bake at 400° F for 8-10 minutes

### Algorithms II

#### Newton-Raphson: roots of \( f(x) = 0 \)

- **INPUT** initial approximation \( p_0 \); tolerance TOL; maximum number of iterations \( N_0 \)
- **OUTPUT** approximate solution \( p \) or message of failure

#### Steps

1. Set \( i = 1 \)
2. While \( i < N_0 \) do steps 3 - 6
   - Set \( p = p_0 - \frac{f(p_0)}{f'(p_0)} \)
   - Step 4 if \( | p - p_0 | < TOL \) then OUTPUT \( p \) and STOP
   - Set \( i = i + 1 \)
   - Set \( p_0 = p \)
3. OUTPUT "Method failed after \( N_0 \) iterations; STOP"

### Flowcharts

#### The basic elements

- Terminator
- Loop preparation
- On-page connector
- Off-page connector
- Control flow
- Process
- Decision

### Algorithms III

#### Newton-Raphson: roots of \( f(x) = 0 \)

- **INPUT** initial approximation \( p_0 \); tolerance TOL; maximum number of iterations \( N_0 \)
- **OUTPUT** approximate solution \( p \) or message of failure

#### Steps

1. Set \( i = 1 \)
2. While \( i < N_0 \) do steps 3 - 6
   - Set \( p = p_0 - \frac{f(p_0)}{f'(p_0)} \)
   - Step 4 if \( | p - p_0 | < TOL \) then OUTPUT \( p \) and STOP
   - Set \( i = i + 1 \)
   - Set \( p_0 = p \)
3. OUTPUT "Method failed after \( N_0 \) iterations; STOP"

### Functional Model

#### The oldest model

- Focuses on how (i.e., the algorithms) to solve a problem
- Decomposes problem into functions and function interfaces
- Changes have wide spread effects – global data makes the program fragile
- Functions have varying degrees of coupling through global data
  - Developed as a unit
  - Debug as a unit
  - Validate as a unit
**Data Driven Models**

- Data flow
  - Maps data input to data output
  - Design data structures first
  - Design processes / functions last
- Data hiding
  - Packages data and the functions that work on the data together in a module (a file in C)
  - Data is still in global scope but access is allowed only through the module functions
- Abstract Data Type (ADT)
  - Programmer created data type
  - struct in C

**Object Model**

- Characteristics of functional & data models
- A tool for managing complexity
- Change resilient
  - Change is localized
  - Intra-object functions may be coupled
  - Extra-object functions are decoupled
- Natural organization for data and functions
  - Objects **encapsulate** data and functions together
  - Supports ADTs: multiple objects of a type may be created (class is a type specifier or ADT)
  - Supports data hiding: data access is controlled through key words

**Software Life Cycle Phases**

**Typical phases**

- Requirements
- Analysis
- Design
- Implementation
- Validation
- Maintenance
- Retire

**Waterfall Model**

IEEE process standard P1074, 1/1/91 (see Fig 1.16, p. 25)

**Spiral Model**

A contemporary life cycle

- Inception
  - idea is sufficiently well founded to warrant entering the elaboration phase
  - strategic/tactical
- Elaboration
  - requirements articulated & prioritized
  - testing planned
- Transition
  - phased delivery
  - move to next component
  - evaluate improvements and corrections
- Construction
  - allocate resources
  - requirements & evaluation criteria examined
  - Code
Design and Development
Details on pp. 33 - 39

- Analyze the problem
  - Outputs
  - Inputs
  - Formulas
  - Perform hand calculation
- Select / develop algorithm
- Write program
- Test and correct

Common Errors
See pp. 37 - 38

- Rushing to code
- Forgetting to back up program
- Failing to understand that computers do exactly as told
- Programming error classifications
  - Syntax
  - Run-time
  - Logical