COMS W4115
Programming Languages and Translators
Lecture 2: January 28, 2013
Basic Elements of Languages and Compilers

Overview

1. Course project
2. What's in a language specification?
3. The C programming language
4. Fundamental elements of programming languages
5. Language processing tools

1. Course Project

- Project description
  - Form a team of five. Contact Maria Taku (mat2185@columbia.edu) for help finding or forming a team. Once your team is complete, let Maria know who is on your team.
  - Create an innovative little language of your own design.
  - Write a compiler or interpreter for your language.
  - Each team member must write at least 500 lines of code of the compiler or interpreter.
  - Project constitutes 40% of final grade.
- Project team: elect one person to serve each of the following functions.
  - Project manager
    - This person sets the project schedule, holds weekly meetings with the entire team, maintains the project log, and makes sure the project deliverables get done on time.
  - Language and tools guru
    - This person defines the baseline process to track language changes and maintain the intellectual integrity of the language.
    - This person teaches the team how to use various tools used to build the compiler.
  - System architect
    - This person defines the compiler architecture, modules, and interfaces.
  - System integrator
    - This person defines the system integration environment and makes sure the compiler components work together.
  - Tester and validator
    - This person defines the test suites and executes them to make sure the compiler meets the language specification.
- Project due dates and deliverables:
  - Feb. 27: Language white paper (written by entire team, 3-4 pages).
    - See http://www.oracle.com/technetwork/java/index-136113.html for a sample white paper on Java.
  - Mar. 27: Language tutorial (written by entire team, 15-20 pages).
    - Chapter 1 of Kernighan and Ritchie is a good model of a language tutorial.
    - Describe a few representative programs that illustrate the nature and scope of your language.
    - A "hello, world" program is de rigueur.
  - Mar. 27: Language reference manual (written by entire team, 20-25 pages).
    - Appendix A of Kernighan and Ritchie is a good model.
    - Give a complete description of the lexical and syntactic structure of your language.
    - Include a full grammar for your language.
  - May 14-16: Working compiler and demo.
  - May 14-16: Final project report due at project demo.

2. A Language Specification Defines

- the representation of programs
- the syntax and constraints of the language
- the semantic rules for interpreting programs
- the representation of input data to be processed by programs
- the representation of output data produced by programs
- other restrictions on programs (such as what makes a program portable across different implementations and platforms)

3. The C Programming Language

- C is a general-purpose procedural programming language that was designed in 1969-1972 at Bell Labs by Dennis Ritchie who was working on developing the Unix operating system with Ken Thompson. It is still one of the most widely used programming languages in the world.
- C was originally designed for and implemented on the UNIX operating system on the DEC PDP-11.
- In 1978 Brian Kernighan and Dennis Ritchie published the book "The C Programming Language" which for many years served as the informal definition of C ("K&R C").
- Kernighan and Ritchie described C as "a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators."
- C has gone through a number of versions since K&R C:
  - ANSI C (1989) and ISO C (1990): these versions are identical and are commonly referred to as C89
  - C99: ISO/IEC9899:1999
  - C11, the current standard, adopted in 12/8/2011
  - Embedded C: in 2008 the C Standards Committee extended C to create programs to meet the stringent requirements of microcontroller systems

- A sample C program

```
(1)  /* this program computes the greatest common divisor
(2)     of two integers entered on the command line */

(3)  #include <stdio.h>
(4)  #include <stdlib.h>

(5)  int gcd(int m, int n)
(6)  {
(7)    int r;
(8)    while ((r = m % n) != 0) {
(9)      m = n;
(10)      n = r;
(11)   }
(12)   return n;
(13) }

(14) int main(int argc, char *argv[])
(15) {
(16)   int m, n;
(17)   m = atoi(argv[1]);
(18)   n = atoi(argv[2]);
(19)   printf("gcd of %d and %d is %d\n", m, n, gcd(m, n));
(20)   return 0;
(21) }
```

4. Fundamental Elements of Programming Languages

- Programming model
  - The programming model is the model of computation encapsulated into the programming language.
    - For example, C is an imperative language, designed around the von Neumann model of computation.
- Program structure
  - A program typically consists of one or more translation units stored in files.
    - In C, a translation unit is a sequence of function definitions and declarations.
- Character set and lexical conventions
  - Source and target character sets may be different.
    - The character set of C source programs in contained within seven-bit ASCII.
  - A token is a meaningful sequence of characters in a source program.
    - C has six classes of tokens: identifiers, keywords, constants, string literals, operators, and separators.
- Names, scopes, bindings, and lifetimes
  - Names (often called identifiers) have a specified lexical structure.
    - In C identifiers are sequences of letters (here, underscore is considered a letter) and digits. The first character of an identifier must be a letter. At least the first 31 characters in an identifier are significant.
  - The scope of a name is the region of the program in which it is known (visible).
  - A binding is an association between two things such as between a variable and its type or between a symbol and the operation it represents. The time at which this association is determined is called the binding time. Bindings can take place at various times ranging from language design time to run time.
  - The lifetime of a variable is the time during which the variable is bound to a specific memory location.
- Memory management

    - One important function of a programming language is to provide facilities for managing memory and the objects stored in memory.
        - C provides three ways to allocate memory for objects:
            - Static allocation where space for an object is provided by the compiler at run time.
            - Automatic allocation where temporary objects are stored by the compiler on the runtime stack. This space is automatically freed by the compiler after the block in which the object is declared is exited.
            - Dynamic allocation where blocks of memory of arbitrary size can be requested by a programmer using runtime library functions such as malloc from a region of memory called the heap. These blocks persist until freed by a call to a runtime library function such as free.
- Data types and operators
    - A data type defines a set of data values and the operations allowed on those values.
        - C has a small number of basic types, including `char, int, double, float, enum, void`.
        - C has a potentially infinite number of recursively defined derived types such as arrays of objects of some type, functions returning objects of some type, pointers to objects of some type, structures containing a sequence of objects of various types, and unions containing any one of several objects of various types.
        - C has a rich set of arithmetic, relational, logical, and assignment operators.
- Expressions and assignment statements
    - Expressions are the primary means for specifying computations in a programming language.
    - Assignment statements are basic constructs in imperative programming languages. Assignment statements allow the programmer to dynamically change the bindings of values to variables.
- Control flow
    - Flow of control refers to the sequence in which the operations specified in a program are executed at run time. There are flow-of-control issues at many levels such as flow of control within expressions, among statements, and among program units. Most programming languages have control statements and other control structures for controlling the flow of control within a program.
        - C has a variety of flow-of-control constructs such as blocks and control statements such as `if-else, switch, while, for, do-while, break, continue` and `goto`.
- Functions and process abstraction
    - Functions are perhaps the most important building blocks of programs. Functions are often called procedures, subroutines, or subprograms. Functions break large computing tasks into smaller ones and facilitate code reuse. Functions are such an important topic in programming languages that we will talk about them in much more detail later in this course.
- Data abstraction and object orientation
    - Data abstraction in the form of abstract data types was introduced into programming languages after process abstraction. The programming language Simula67 was instrumental in motivating constructs for supporting object-oriented programming in modern programming languages such as C++, C#, and Java. Object orientation is characterized by encapsulation, polymorphism, inheritance, and dynamic binding.
- Concurrency
    - Concurrent execution of programs has assumed much more importance with the widespread use of multi-core and many-core processors.
    - Concurrency in software execution can occur many levels of granularity: instruction, statement, subprogram, and program.
    - Concurrency can be achieved with libraries (like MPI for Fortran, pthreads for C) or with direct language support (as in Cilk, X10).
    - However, effective exploitation of concurrency is still an open research area in software.
- Exception and event handling
    - Many languages have facilities for reacting to run-time error conditions. C++ has the `try-catch` construct to catch exceptions raised by the `throw` statement.
    - Event handling is like exception handling in that an event handler is called by the occurence of an event. Implementing reactions to user interactions with GUI components is a common application of event handling.

5. Language Processing Tools

- Basic compiler
- Interpreter
- Bytecode interpreter
- Just-in-time compiler
- Linker and loader
- Preprocessor

6. Practice Problems

1. Describe the von Neumann model of computation (computer architecture).
2. Compare C and Java with regard to their (a) programming model and (b) primitive data types.

7. Reading Assignment

- ALSU: Chapters 1 and 2

8. Reference

- Brian Kernighan and Dennis Ritchie, *The C Programming Language*, 2nd edition, Prentice Hall, 1988. This is the classic reference on ANSI C (C89).

---

aho@cs.columbia.edu