CS 3710
Vocational Languages
September 5, 2012

Introduction

• Course Website:
• Schedule:
• Assignment: online submission + hardcopy at class
• Introduce yourself:
  – C?
  – C++?
  – JavaScript?
  – Scala?

C Language History

• Developed at Bell Laboratories in 1972 by Dennis Ritchie.
• It is a high-level language (vs. low-level language)
• It is a compiler language (vs. interpreted language).
• It is a procedure language (vs. object-oriented language).
• There were various versions but in 1983, the American National Standard Institute (ANSI) formed a committee to standardize the language: ANSI C.

C Language Features

• High-level language syntax:
  – Human-friendly, flexible
  – Like Fortran and Pascal, used for general-purpose programming
• Low-level commands:
  – Bitwise operators, pointer manipulation capability
  – Machine-friendly, fast
  – Like Assembly language, used for system programming.
The C Program Development Cycle

- **Edit Program**
- **Source Code**
- **Compile**
- **Object Code**
- **Link Object Code**
- **Executable**

**Example**

- The standard I/O header:
  ```c
  #include<stdio.h>
  */
  ```
- main returns 0 when the program completes successfully.
- Java main does not return anything (void).

```c
/* hello world program hello.c */
#include<stdio.h>
main()
{
  printf("Hello World!\n");
}
```

**Compile and Execute C Programs**

- Compilation translates your source code (in the file hello.c) into object code (machine dependent instructions for the particular machine you are on).
  - Note the difference with Java:
    - The javac compiler creates Java byte code from your Java program.
    - The byte code is then executed by a Java virtual machine, so it's machine independent.
- Linking the object code will generate an executable file.
- There are many compilers for C under Unix
  - SUN provides the Workshop C Compiler, which you run with the cc command
  - There is also the freeware GNU compiler gcc

**Compile and Execute C Programs**

- To compile a program:
  - Compile the program to object code.
    ```
    > cc –c hello.c
    ```
  - Link the object code to executable file.
    ```
    > cc hello.o –o hello
    ```
- You can do the two steps together by running:
  ```
  > cc hello.c –o hello
  ```
- To run your program:
  ```
  > ./hello
  Hello World!
  ```

If you leave off the -o, executable goes into the file a.out
Variable Names

- Variables and constants are the basic data objects manipulated in a program.
- Variable names may contain letters, digits, and underscores.
- The first character must be a letter or an underscore.
- Case sensitive: All uppercase variables are for constants.
- C keywords cannot be used as variable names.

Basic Types

- Variables store values, which can have the following basic data types:
  - char: a single byte, capable of holding one character, e.g. ‘a’, ‘b’...
  - int: an integer of fixed length, typically reflecting the natural size of integers on the host machine (i.e., 32 or 64 bits)
  - float: single-precision floating point
  - double: double precision floating point

Basic Types - Continue

- There are a number of qualifiers which can be applied to the basic types
  - length of data:
    - short int: “shorter” int, which should be <= the number of bits in an int
    - long int: “longer” int, which should be >= the number of bits in an int
    - long double: generally extended precision floating point
    - signed and unsigned:
      - signed int: an int type with no sign (default of int is signed)
      - unsigned int: a number from 0 to 2^31-1
      - signed char: a number from 0 to 255 (chars are 8 bits, 0-255)
      - unsigned char: a number from 0 to 255 (chars are 8 bits, 0-255)
  - The sizes of these values are stored in `<limits.h>`

Constants

- You can also declare variables as being constants.
  - Use the const qualifier:
    - const long double pi=3.1415926l;
    - const float max1=1.0f;
    - const int val=(3*7+6)*5;
    - const char first='a';
  - These are an older form of constant declaration
  - #define CONSTNAME literal
    - Generally make pre-processor constants all upper case (convention).
  - Example:
    - define PI 3.14159
  - What really happens:
    - The C preprocessor runs before the compiler.
    - Every time it sees the token PI, it substitutes the value 3.14159.
    - The compiler is then run with this "pre-processed" C code.

Note: Simple computed values are allowed
  - must be able to evaluate at compile time
Variables Declarations

- All variables must be declared before use.
- A declaration specifies a type, and contains a list of one or more variables of that type:
  - int lower, upper, step;
  - char c, line[1000];
- Where declarations appear affects their scope and visibility
  - Rules are similar to those in Java
  - Declaration outside of any function are for global variables
    - e.g., just before the main routine

Variable Initialization

- A variable may also be initialized in its declaration.
- Examples:
  ```
  int count = 0;
  ```
- Assign a value to a variable using =:
  ```
  int count; /* declaration*/
  count = 0; /* assign a value */
  ```
- An uninitialized/unassigned variable has undefined (i.e. garbage) value:
  ```
  int count; printf("%d", count);
  // no compile time error, print garbage
  // Java compiler would report an error at compile time
  ```