ECE 131 – Programming Fundamentals

Module 2, Lecture 1: Data Types, Variables and Expressions – Introduction

Dr. Daryl Lee

University of New Mexico

Module Overview

- The history of C
- The C programming model
- The anatomy of a C program
- C syntax –
  - Comments
  - Number Systems
  - Data Types
  - Declaring Variables and Constants
  - Expressions and Statements

History of C

- The C programming language was created at AT&T Bell Laboratories in the early 1970's as the successor to the B programming language.
- C was used to develop the UNIX operating system at AT&T Bell Laboratories, and the language itself has always been packaged as part of a standard UNIX distribution.
- Early C compilers were based on the definition provided in the appendix of the text: *The C Programming Language*, by Brian Kernighan and Dennis Ritchie. To this date, that version of the language is referred to as K&R C.
- One of the major advantages of C has always been its portability.
- The K&R specification was not complete, and contained ambiguities. Thus, as the language became more popular, and more vendors began to create C compilers, portability suffered. I.e., code intended for one C compiler might not compile using a different C compiler.

History of C

- Other incompatibilities arose when vendors added special language features to their compilers.
- In order to address this problem, the American National Standards Institute (ANSI) initiated an effort to standardize C in the early 1980's.
- In 1990, the ANSI C stardard was released.
- The International Standards Institute (ISO) adopted the ANSI C standard, naming it ISO/IEC 9899:1990.
- Since that time, these standards committees have continued to review and update the C language, and the most recent standard, released in 1999, is referred to as the ANSI C99 standard, or ISO/IEC 9899:1999.

Basic Programming Language Elements

The programming model for C the imperative paradigm (procedure-oriented). In addition, with C, your model includes an underlying von Neumann architecture. Thus, we need to know:

- How to specify procedures (subroutines, functions in the C "world").
- What are the available data types.
- The rules for declaring variables and constants.
- The available control structures.
- Other language features.

The remainder of this class will be concerned with these topics.

Procedure-oriented Paradigm in C

```
/* function prototypes */
float func1(float);
int func2(int);

main()
{
   float x;   // floating-point variable x
   float y=2.2;   // floating-point variable y
   x = func1(y);
   printf("\n\n x =  %f \n\n", x);
}

float func1(float num)
{
   int x=2;  // a different integer variable x
   x = func2(x);
   return (x * num);
}

int func2(int val)
{
    return (val * 2);
}
```

Anatomy of a C Program

- Every C program has the following structure:

  declarations – variables, constants, functions
  ```
  int main(void)
  {
      program statements
  }
  ```
- Every C program must have a function called `main()`, and this is where the execution of the C program always starts.
- The "void" in the function definition is for documentation purposes only. It means "I thought about this and no parameters are needed."
- The open brace "{" and close brace "}" denote the beginning and end of the `main()` function, respectively. "{" and "}" are more concise than "begin" and "end".
- The program statements (or simply statements) inside `main()` may involve calls to other functions.

Comments

- A comment is a programming language construct that allows information to be embedded into the source code of a program.
- Comments do not produce executable code, i.e., the compiler/interpreter ignores them.
- Why use comments?
  - To annotate code with descriptions so that at a later date, you (or other programmers) can more easily understand the logic behind the code. I.e., the annotations might clarify the intent of the code, explain what the programming was thinking at the time the code was developed, serve as a placeholder for where additional code should be developed, etc.
  - To embed logos, copyright and other identifying information in source code.
  - Comment-based debugging — comment out sections of source code during debugging that are causing a program not to compile.

Comments

- The trend in software development is to write code that is very descriptive, this is often referred to as self-commenting code.
  Ex:

  ```
  circumference = PI * diameter;
  ```

  is far better than

  ```
  axz = 3.14 * dd8;
  ```

  although both are valid C statements that could be calculating the same thing. The latter is more likely to require a comment.
- Increasingly, comments are being processed by various external programming tools.
  - Automatic generation of external documentation.
  - Some source code management systems and software modeling tools use comments to manage versions.

Comments in C

```
/* function prototypes */
float func1(float);
int func2(int);

main()
{
   float x;   // floating-point variable x
   float y=2.2;   // floating-point variable y
   x = func1(y);
   printf("\n\n x =  %f \n\n", x);
}

float func1(float num)
{
   int x=2;  // a different integer variable x
   x = func2(x);
   return (x * num);
}

int func2(int val)
{
    return (val * 2);
}
```

THE UNIVERSITY of NEW MEXICO

Comments in C

- A comment can appear anywhere in a C program where a blank space is allowed. There are two ways to insert comments in a C program:
  1. A comment can begin with the two characters //. In this case, any characters that follow // on that line will be treated as a comment by the compiler.
  2. A comment can also begin with the two characters /∗, and end when the two characters ∗/. This type of comment can extend over more than one line. These types of comments cannot be nested.
     - This style of commenting is useful during debugging, as it allows you to quickly comment out multiple lines of code.
     - If some of the lines you're commenting out contain comments in the style `/* comments */`, the first ∗/ encountered will end the comment, ignoring any other /∗ characters encountered after the first /∗ that began the comment.

Comments in C

```
#include <stdio.h>
main()
{
   int z, z_factorial;

   z = 5
   z_factorial = z!;
   printf("\n z factorial = %i \n", z_factorial);
}
```

- This program does not compile.

Comments in C

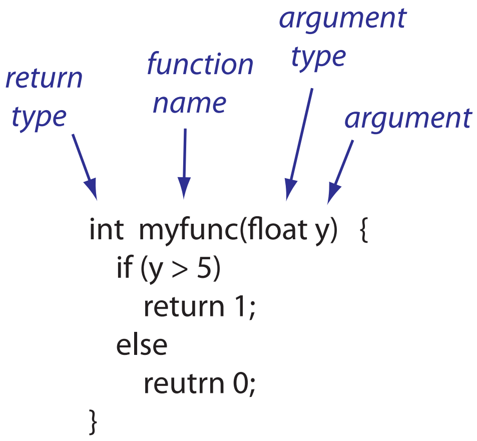
```
#include <stdio.h>
main()
{
   int z, z_factorial;

   /*
   z = 5
   z_factorial = z!;
   */
   printf("\n z factorial = %i \n", z_factorial);
}
```

- By “commenting them out”, we know there is a syntax problem in these two lines.

THE UNIVERSITY of NEW MEXICO

Functions in C – A First Look



```
int  myfunc(float y)   {
   if (y > 5)
      return 1;
   else
      reutrn 0;
}
```