C Programming Basics

SDS 322/329

October 15, 2015

Email any questions to:
rauta@tacc.utexas.edu

Administrative Trivia

- **Quiz-3** will be held at the beginning of next class and will be based on the topics covered in the C lectures

Recap & Today’s Lecture

- In the previous class
  - Understanding a basic C program
    - Printing to standard output (screen)
    - Compiling and running a C program
- Today’s Lecture
  - Understanding Errors
  - Comments: two types – single line (//) or multi-line (/* */)
  - Keywords: reserved words – e.g., return, int
  - Variables: declaration and initialization
  - Data Types and Identifiers

Overview of Content

All the concepts are accompanied by examples.

Warnings, Errors and Bugs

- Compile-time warnings
  - Diagnostic messages
- Compile-time errors
  - Typographical errors: pirntf , $include
- Link-time errors
  - Missing modules or library files
- Run-time errors
  - Null pointer assignment
- Bugs
  - Unintentional functionality

Find the Error: error.c

```
#include <stdio.h>
int main(){
  printf("Find the error!\n")
  retrun(0);
}
```

Error Message (compile-time error)

```
**** Internal Builder is used for build****
gcc -O0 -g3 -Wall -c -fmessage-length=0 -oerror.o ..
\error.c
..\error.c: In function 'main':
..\error.c:4:3: error: expected ';' before 'retrun'
..\error.c:5:1: warning: control reaches end of non-
void function
Build error occurred, build is stopped
Time consumed: 148  ms.
```

Find the Error: error.c

```
#include <stdio.h>
int main(){
  printf("Find the error!\n");
  retrun 0;
}
```

Error Message (link-time error)

```
gcc -o error error.c
…
..\error.c:4:3: warning: implicit declaration of
function 'retrun'
…
gcc -oCTraining.exe error.o
error.o: In function `main':
C:\Users\ra25572\workspace\CTraining\Debug/../error.c:4:
undefined reference to `retrun'
collect2: ld returned 1 exit status
Build error occurred, build is stopped
Time consumed: 436  ms.
```

Find the Error: error2.c

```
#include < stdio.h >
int main(){
  printf("Find the error!\n");
  return 0;
}
```

Error Message (compile-time error)

```
gcc -o error2 error2.c
..\error2.c:1:21: fatal error:  stdio.h : No
such file or directory
compilation terminated.
Build error occurred, build is stopped
Time consumed: 98  ms.
```

Overview of Content

- Writing a Basic C Program
- Understanding Errors
- Comments, Keywords, Identifiers, Variables
- Operators
- Standard Input and Output
- Control Structures
- Functions in C
- Arrays, Structures
- Pointers
- Working with Files

All the concepts are accompanied by examples.

Comments and New Line: rules.c

```c
/*
 * rules.c
 * this is a multi-line comment
 */
#include <stdio.h>
int main(){
  printf("Braces come in pairs.");
  printf("Comment tokens come in pairs.");
  printf("All statements end with semicolon.");
  printf("Every program has a main function.");
  printf("C is done mostly in lower-case.");
  return 0;
}
```

Output of rules.c

```
Braces come in pairs.Comment tokens come in
pairs.All statements end with a
semicolon.Every program must have a main
function.C is done mostly in lower-case.
```

Output looks odd! We want to see a new line of text for every `printf` statement.

Comments and New Line: rules.c

```c
/*
 * rules.c
 * this is a multi-line comment
*/
#include <stdio.h>
int main(){
 /* notice the \n in the print statements */
 printf("Braces come in pairs.\n");
 printf("Comment tokens come in pairs.\n");
 printf("All statements end with semicolon.\n");
 printf("Every program has a main function.\n");
 printf("C is done mostly in lower-case.\n");
 return 0;
}
// this is another way to specify single-line comments
```

Output of rules.c

```
Braces come in pairs.
Comment tokens come in pairs.
All statements end with a semicolon.
Every program must have a main function.
C is done mostly in lower-case.
```

The output looks better now!

Do-It-Yourself Activity

- Learn the various ways in which you can print and format values of various data types.
- For example:
  - How would you print an integer?
    - See an example of this on slide #24
  - How would you print a value of type double with precision of 8 places after the decimal?
- Reference:
  - http://www.cplusplus.com/reference/clibrary/cstdio/printf/

Some C Language Keywords

Category	Keywords
Storage class specifiers	`auto register static extern typedef`
Structure & union specifiers	`struct union`
Enumerations	`enum`
Type-Specifiers	`char double float int long short signed unsigned void`
Type-Qualifiers	`const volatile`
Control structures	`if else do while for break continue switch case default return goto`
Operator	`sizeof`
Deprecated keywords	`fortran entry`
Other reserved words	`asm bool friend inline`

Variables

- Information-storage places
- Compiler makes room for them in the computer's memory
- Can contain string, characters, numbers *etc.*
- Their values can change during program execution
- All variables must be declared before they are used and must have a data type associated with them
- Variable must be initialized before they are used

Data Types

- Data types specify the type of data that a variable holds

- Categories of data types are:
  - **`Built-in:`** **`char double float void int (short long signed unsigned)`**
  - **`User-defined:`** **`struct union enum`**
  - **`Derived: array function pointer`**

- We have already seen an example code in which an integer data type was used to return a value from a function:

  ```
  int main()
  ```

- Compiler-dependent range of values associated with each type. For example: an **`int`** can have a value in the range
  –32768 to **32767** on a 16-bit computer or
  –2147483647 to **2147483647** on a 32-bit computer

Identifiers

- Each variable needs an identifier (or a name) that distinguishes it from other variables
- A valid identifier is a sequence of one or more letters, digits or underscore characters
  - Note: you cannot begin the variable name with a digit
- Keywords cannot be used as identifiers

Variable Declaration

- Declaration is a statement that defines a variable
- Variable declaration includes the specification of data type and an identifier. Example:

```
int number1;
float number2;
```

- Multiple variables can be declared in the same statement

```
int x, y, z;
```

- Some types of data can be signed or unsigned
- Signed types can represent both positive and negative values, whereas unsigned types can only represent positive values

```
signed double temperature;
```

Variable Initialization

- A variable can be assigned a value when declared
  - Assignment operator is used for this purpose
  - `int x = 10;`

- More examples
  - `char x = 'a';`
  - `double x = 22250738585072014.e23;`
  - `float x = 10.11;`

- `void` cannot be used to declare a regular variable
  - It is used as a return type of a function or as an argument of a function

Example of Updating Variables: myAge.c

```c
#include <stdio.h>
int main(){
  int age;
  age = 10;
  printf("Initial value of age is: %d\n", age);
  age = 20;
  printf("Updated value of age is: %d\n", age);
  age = age + 20;
  printf("New updated value of age is: %d\n", age);
  return 0;
}
```

Output:

```
Initial value of age is: 10
Updated value of age is: 20
New updated value of age is: 40
```

Casting between variables

- By type-casting, a variable can be temporarily made to look like another variable
- To typecast a variable of an already defined data-type, put the new data-type that you temporarily want inside parentheses in front of it
  - see next slide for example

Type-casting double as char: myTypeCast1.c

```c
#include <stdio.h>
int main(){
  double varA = 65.00;
  char varB;
  varB = (char) varA;
  printf("varA: %lf, varB: %c",varA, varB);
  return 0;
}
```

Output:

```
varA: 65.000000, varB: A
```

References

- C Programming Language, Brian Kernighan and Dennis Ritchie
- Let Us C, Yashavant Kanetkar
- C for Dummies, Dan Gookin
- http://cplusplus.com
- http://www.cprogramming.com/tutorial/c/lesson11.html