Programming Languages Lab

Bibhas Adhikari

IIT Kharagpur

September 4, 2020

A brief review of C programming language

- C is a language developed in 1972 by Dennis Ritchie and Ken Thompson of AT&T Bell Telephone Laboratories.
- It is related to Algol and Pascal in style, with attributes of PL/I thrown in. Although a general-purpose programming language, its compact syntax and efficient execution characteristics have made it popular as a systems programming language.
- C is associated with systems programming activities. It was first used to write the kernel of the UNIX operating system and has been closely tied to UNIX implementations every since, although versions of C exist on most computing systems today.

Fundamentals

- The set of basic operations that a computer system can perform is called the computer's instruction set.
- A computer program is just a collection of the instructions necessary to solve a specific problem
- A method that is used to solve the problem is known as an algorithm
- Higher-level language: High-level language, assembly language, machine instructions
- Compiler: A computer program which translates the statements of the program developed in a higher-level language into a form that the computer can understand
- Operating system: It is a program that controls the entire operation of a computer system. All input and output (that is, I/O) operations that we perform on a computer system are channeled through the operating system (UNIX, Linux, Mac OS X, Microsoft Windows XP)

Installing an environment

1. The minimum requirements for learning how to program in C are access to a computer, a text editor, C libraries, and a C compiler.
2. There are a number of free C compilers and text editors that you can use and, of course, there are many more that cost money
3. Cygwin have cleverly developed a simple, yet robust Linux-like environment for Win-tel (Microsoft Windows–Intel) platforms that includes many free software packages, such as a C compiler called gcc, text editors, and other common utilities.
4. You can download Cygwin's free software components from their website at http://www.Cygwin.com.
5. The Cygwin setup process is very easy, but if you have questions or issues you can visit the online user guide via http://cygwin.com/cygwin-ug-net/cygwin-ug-net.html

1. Run the setup file directly from Cygwin's website (http://www.cygwin.com/setup.exe).
2. After successfully installing the Cygwin environment, you will have access to a simulated UNIX operating system through a UNIX shell
3. The first line shows that you are logged into the UNIX shell as Administrator (default login name) at your computer (Do IT YOURSELF)
4. The next line starts with a dollar sign ($).
5. Depending on your specific installation (Cygwin version) and configuration (components selected) of Cygwin, you may need to have Cygwin's bin directory, added to your system's *PATH* environment variable.

Help1: https://www.staff.ncl.ac.uk/andrey.mokhov/EEE1008/install-cygwin.html
Help2: https://www.staff.ncl.ac.uk/andrey.mokhov/EEE1008/first-c-program.html

CodeBlocks

1. Download binary: codeblocks-10.05mingw-setup.exe from http://www.codeblocks.org/downloads/26
2. Save the file at the Desktop. Download of 70+ MB file will take some time.
3. Run codeblocks-20.03mingw-setup.exe : Next, I agree, Next, Install
4. Run Code::block, select compiler
5. Settings: compiler and debugger - it will show you GNU GCC Compiler.
6. My Computer, Local Disk (C:), Program Files, CodeBlocks, MinGW, bin
7. My Computer, View System Information, Advanced, Environment Variables,
8. If PATH is there, append it with the path of C:\Program Files\CodeBlocks\MinGW\bin otherwise create PATH vaiable and put the path as the value.

Compiling a program

- A compiler is a software program which analyzes a program developed in a particular computer language and then translates it into a form that is suitable for execution on your particular computer system
- The program that is to be compiled is first typed into a file, called source program on the computer system. In general, the choice of the name is up to you. C programs can typically be given any name provided the last two characters are "*.c*"
- Compilation process:
  - The compiler examines each program statement contained in the source program and checks it to ensure that it conforms to the syntax and semantics of the language
  - any mistakes are detected by the compiler then it is refereed to the user and the process ends
  - Then the errors should be corrected and the new source program is to be compiled again

Compiling a program (cont.)

- Compilation process (cont.)
  - Once the source program is error free then it is translated into an equivalent assembly language program
  - Then the assembly language statements are converted into actual machine instructions (assembler, however in most computer systems this is an automatic process)
  - Then the assembly language statements are converted into binary format called object code, which is written into another file on the computer system
  - Other subprograms used in the source program and the programs in the system's program library are also searched and linked together with the object program
  - This final file will be executed by the computer
  - If the desired output is not produced by the computer in console then it needs to be debugged (debugging process) to remove the logical errors in the program

Integrated Development Environments and Language Interpreters

IDE

The process of editing, compiling, running, and debugging programs is managed by an integrated application called Integrated Development Environment (IDE).

- Windows - Microsoft Visual Studio, ...
- Linux - Kylix,....
- Mac OS X - CodeWarrior, Xcode, ...

Interpreters

Interpreted but not compiled by a compiler

- BASIC, JAVA, *shell*, Python

main() FUNCTION

1. We start with the beginning of every C program, the *main()* function.
2. From a programming perspective, functions allow you to group a logical series of activities, or program statements, under one name.
3. suppose we want to create a function called *makeTea*. My algorithm for making tea might look like this:

   Add 1 cup/200 mL of freshly boiled water to your tea bag (in a mug)
   Allow the tea bag to brew for 2 minutes
   Remove the tea bag
   Add 10 mL of milk
4. Functions are typically not static, meaning they are living and breathing entities, again philosophically, that take in and pass back information. Thus, my *makeTea* function would take in a list of ingredients (called parameters) and return back a finished Tea (called a value).

Algorithm

An algorithm is a finite step-by-step process for solving a problem. It can be as simple as a recipe to making tea, or as complicated as the process to implement an autopilot system

- Algorithms generally start off with a problem statement
- It is this problem statement that programmers use to formulate the process for solving the problem
- Building algorithms and algorithm analysis occurs before any program code has been written

main () function

We will use *main()* functions that are void of parameters (functions that do not take parameters) and do not return values.

```
main()
{
}
```

The first brace denotes the beginning of a logical programming block and the last brace denotes the end of a logical programming block. Each function implementation requires that you use a beginning brace, {, and a closing brace, }.

CAUTION

C is a case-sensitive programming language. For example, the function names *main(), Main()*, and *MAIN()* are not the same. It takes extra computing resources to NOT be case-sensitive as input devices such as keyboards distinguish between cases.

Building blocks of a C-program

/ C programming */* (multi-line comment block)
// by BA (single line comment block)
 # include <stdio.h> (preprocessor directive)(standard input output library)
main() (function)
{ (begin logical program block)
printf("\nC you later \n"); (function) (escape sequence) (program statement terminator) – (program statement)
} (end logical program block)

Comment

1. `Comments` are an integral part of program code in any programming language. `Comments` help to identify program purpose and explain complex routines.
2. In the above program, *C programming* is ignored by the compiler because it is surrounded with the character sets /* and */.
3. The character set /* signifies the beginning of a comment block; the character set */ identifies the end of a comment block.
4. Any characters read after the character set // are ignored by the compiler for that line only.
5. To create a multi-line comment block with character set //, you will need the comment characters in front of each line.

Keywords in ANSI C

Keyword	Description
auto	Defines a local variable as having a local lifetime
break	Passes control out of the programming construct
case	Branch control
char	Basic data type
const	Unmodifiable value
continue	Passes control to loop's beginning
default	Branch control
do	Do While loop
double	Floating-point data type
else	Conditional statement
enum	Defines a group of constants of type *int*
extern	Indicates an identifier as defined elsewhere
float	Floating-point data type
for	For loop
goto	Transfers program control unconditionally
if	Conditional statement

Keywords in ANSI C

Keyword	Description
int	Basic data type
long	Type modifier
register	Stores the declared variable in a CPU register
return	Exits the function
short	Type modifier
signed	Type modifier
sizeof	Returns expression or type size
static	Preserves variable value after its scope ends
struct	Groups variables into a single record
switch	Branch control
typedef	Creates a new type
union	Groups variables that occupy the same storage space
unsigned	Type modifier
void	Empty data type
volatile	Allows a variable to be changed by a background routine
while	Repeats program execution while the condition is true

Program Statements

1. Many lines in C programs are considered program statements, which serve to control program execution and functionality.
2. Many of these program statements must end with a statement terminator. Statement terminators are simply semicolons (;).
3. the *printf()* function, demonstrates a program statement with a statement terminator.
4. Some common program statements that do not require the use of statement terminators are the following:
   - Comments
   - Preprocessor directives (for example, *#include* or *#define*)
   - Begin and end program block identifiers
   - Function definition beginnings (for example, *main()*)

Escape character or escape sequence

Escape sequences are specially sequenced characters used to format output.

1. The backslash character (\) is the escape character.
2. When the *printf()* statement shown above is executed, the program looks forward to the next character that follows the backslash.
3. the next character is the character *n*. Together, the backslash (\) and *n* characters make up an escape sequence.
4. This particular escape sequence (*n*) tells the program to add a new line.

COMMON ESCAPE SEQUENCES

Escape sequence	purpose
\n	Creates a new line
\t	Moves the cursor to the next tab
\r	Moves the cursor to the beginning of the current line
\\	Inserts a backslash
\"	Inserts a double quote
\'	Inserts a single quote

homework

- `printf("line 1\n line2 \n line3\n");`
- write a program using escape sequence such that output becomes the calendar of a month

Directive

Notice the program statement that begins with the pound sign (#):
`# include <stdio.h>`

1. When the C preprocessor encounters the pound sign, it performs certain actions depending on the directive that occurs prior to compiling.
2. In the preceding example, the preprocessor is asked to include the `stdio.h` library with the program.
3. The name `stdio.h` is short for standard input output header file
4. It contains links to various standard C *library functions*, such as `printf()`
5. Excluding this preprocessor directive will not have an adverse affect when compiling or running your program. However, including the header file allows the compiler to better help you determine error locations
6. Thus always add a directive to include any library header files that you use in your C programs.