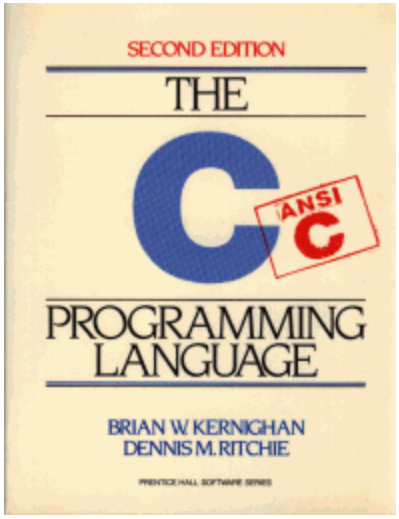
Design and Analysis of Algorithms 214.

Introduction to the C Programming Language.

U.U.Samantha Rajapaksha
BSc. Eng. (Moratuwa), MSc in IT
Senior Lecturer
Sri Lanka Institute of Information Technology
New Kandy Road,
Malabe, Sri Lanka
Tel:0112-301904
email: samantha.r@sliit.lk
Web: www.sliit.lk



C History

- The C language was first developed in 1972 by Dennis Ritchie at AT&T Bell Labs. Ritchie called his newly developed language C simply because there was a B programming language already. (As a matter of fact, the B language led to the development of C.)
- C is a high-level programming language. In fact, C is one of the most popular general-purpose programming languages.
- The first version of Unix was written in the low-level PDP-7 assembler language.
- Ken Thompson developed a compiler for a new high-level language called B, based on the earlier BCPL language developed by Martin Richard.

C History (Contd.)

- BCPL is a simple typeless language .
- BCPL (Basic Combined Programming Language) is a computer programming language designed by Martin Richards of the University of Cambridge in 1966.
- Originally intended for writing compilers for other languages, BCPL is no longer in common use. However, its influence is still felt due to its role in inspiring Dennis Ritchie's widely-used C programming language.
- BCPL was the first curly bracket programming language.
- The language is unusual in having only one data type: a word, a fixed number of bits,
- B was used for further development of the Unix system, which made the work much faster and more convenient.

C History (Contd.)

- When the PDP-11 computer arrived at Bell Labs, Dennis Ritchie built on B to create a new language called C which inherited Ken Thompson's taste for concise syntax, and had a powerful mix of high-level functionality and the detailed features required to program an operating system. Most of the components of Unix were eventually rewritten in C.

The C Language

- Currently, the most commonly-used language for embedded systems
- “High-level assembly”
- Very portable: compilers exist for virtually every processor
- Easy-to-understand compilation
- Produces efficient code
- Designed for systems programming
  - Operating systems
  - Utility programs
  - Compilers
  - Filters

Standard C.

- Standardized in 1989 by ANSI (American National Standards Institute) known as ANSI C
- International standard (ISO) in 1990 which was adopted by ANSI and is known as C89
- As part of the normal evolution process the standard was updated in 1995 (C95) and 1999 (C99)
- C++ and C
  - C++ extends C to include support for Object Oriented Programming and other features that facilitate large software development projects.

Writing C Programs

- A programmer uses a **text editor** to create or modify files containing C code.
- Code is also known as **source code**.
- A file containing source code is called a **source file**.
- After a C source file has been created, the programmer must **invoke the C compiler** before the program can be **executed** (**run**).
- we will use the gcc compiler as it is the compiler available on the Linux system.
- All grading is down on Linux. If you use any other compiler, run it first on the system where the grading is being done. There are differences between the compilers and we only support the gcc compiler on Linux.

Invoking the gcc Compiler

At the prompt, type

- gcc -ansi -Wall pgm.c
- where pgm.c is the C program source file.

- -ansi is a compiler option that tells the compiler to adhere to the ANSI C standard.-Wall is an option to turn on all compiler warnings (best for new programmers).
- If there are no errors in pgm.c, this command produces an executable file, which is one that can be executed (run).
- The gcc compiler names the executable file a.out .
- To execute the program, at the prompt, type

  a.out
- Although we call this process “compiling a program,” what actually happens is more complicated.
- Create example file: try.c
- Compile using gcc:

  gcc –o try try.c
- Execute program

  ./try

Stages of Compilation.

Stage 1: **Preprocessing**

- Performed by a program called the **preprocessor**.
- Modifies the source code (in RAM) according to **preprocessor directives (preprocessor commands**) embedded in the source code.
- Strips comments and white space from the code.
- The source code as stored on disk is <u>not</u> modified.

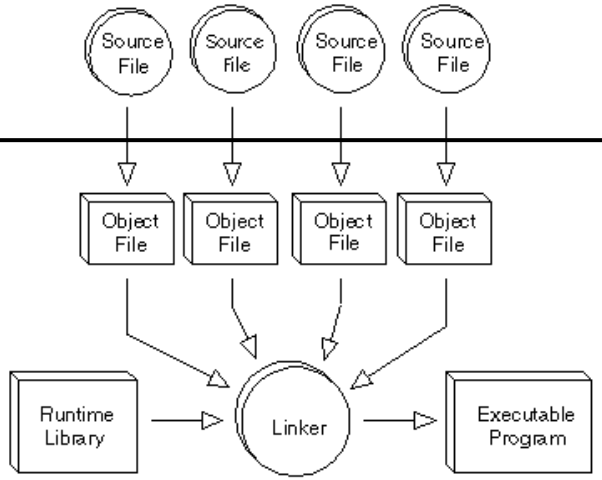
Stages of Compilation. (contd.)

Stage 2: **Compilation**

- Performed by a program called the compiler
- Translates the preprocessor-modified source code into object code (machine code)
- Checks for syntax errors and warnings
- Saves the object code to a disk file, if instructed to do so (we will not do this).
  - If any compiler errors are received, no object code file will be generated.
  - An object code file will be generated if only warnings, not errors, are received.

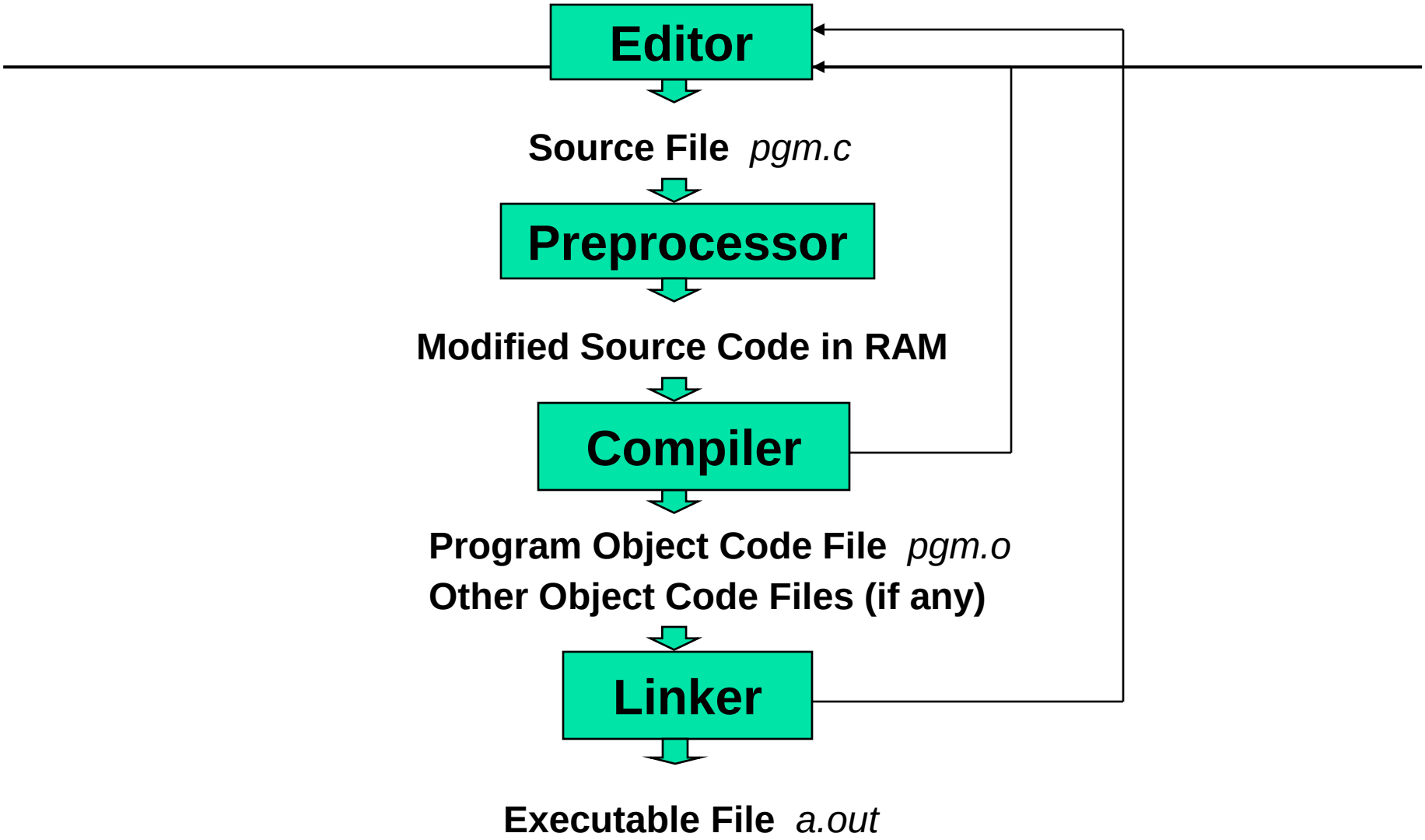
Stages of Compilation. (contd.)

Stage 3: **Linking**



- Combines the program object code with other object code to produce the executable file.
- The other object code can come from the Run-Time Library, other libraries, or object files that you have created.
- Saves the executable code to a disk file. On the Linux system, that file is called a.out.
  - If any linker errors are received, no executable file will be generated.

Program Development Using gcc.



A Simple C Program

```
/*
 ********************************************
 *** Program: hello_world                 ***
 *** Author: Samantha(samantha.r@sliit.lk) ***
 *** Course: DAA 214 Semester two 2012     ***
 *** Lab: Lab 01. Fridays 10:30am          ***
 *** Description: Prints the sentence      ***
 ***   "Hello, world!" to standard output.  ***
 ********************************************
 */
  #include  <stdio.h>
  int main ()
  {
       printf ( "Hello, World!\n" ) ;
       return 0 ;
  }
```

Program Header Comment and Preprocessor Directives

- A comment is descriptive text used to help a reader of the program understand its content.
- All comments must begin with the characters /* and end with the characters */
- These are called comment delimiters
- The program header comment always comes first.

- Lines that begin with a # in column 1 are called preprocessor directives (commands).
- Example: the #include <stdio.h> directive causes the preprocessor to include a copy of the standard input/output header file stdio.h at this point in the code.
- This header file was included because it contains information about the printf ( ) function that is used in this program.

C Standard Library

- Every implementation of C comes with a standard library of predefined functions.
- Note that, in programming, a library is a collection of functions.
- The functions that are common to all versions of C are known as the C Standard Library.

C Standard Library Function Examples

Function Name	Math Name	Value	Example		
abs(x)	absolute value	$\|x\|$	abs(-1)	returns	1
sqrt(x)	square root	$x^{0.5}$	sqrt(2.0)	returns	1.414…
exp(x)	exponential	e^x	exp(1.0)	returns	2.718…
log(x)	natural logarithm	$\ln x$	log(2.718…)	returns	1.0
log10(x)	common logarithm	$\log x$	log10(100.0)	returns	2.0
sin(x)	sine	$\sin x$	sin(3.14…)	returns	0.0
cos(x)	cosine	$\cos x$	cos(3.14…)	returns	-1.0
tan(x)	tangent	$\tan x$	tan(3.14…)	returns	0.0
ceil(x)	ceiling	$\lceil x \rceil$	ceil(2.5)	returns	3.0
floor(x)	floor	$\lfloor x \rfloor$	floor(2.5)	returns	2.0

stdio.h and int main ()

- When we write our programs, there are libraries of functions to help us so that we do not have to write the same code over and over.
- Some of the functions are very complex and long. Not having to write them ourselves make it easier and faster to write programs.
- Using the functions will also make it easier to learn to program.
- Every program must have a **function** called **main**. This is where program execution begins.
- main() is placed in the source code file as the first function for readability. There must be a function with this name or gcc can not successful compile and link your program.
- The **reserved word** “int” indicates that main() **returns** an integer value.
- The parentheses following the reserved word “main” indicate that it is a function.

The Function Body

- A left brace (curly bracket) -- **{** -- begins the **body** of every function. A corresponding right brace -- **}** -- ends the function body.
  - printf (“Hello, World!\n”) ;
- This line is a C **statement**.
- It is a **call** to the function **printf ( )** with a single **argument (parameter)**, namely the **string** “Hello, World!\n”.
- Even though a string may contain many characters, the string itself should be thought of as a single quantity.
- Notice that this line ends with a semicolon. All statements in C end with a semicolon.

return 0 ;

- Because function main() returns an integer value, there must be a statement that indicates what this value is.
- The statement

  return 0 ;

  indicates that main() returns a value of zero to the operating system.
- A value of 0 indicates that the program successfully terminated execution.
- Do not worry about this concept now. Just remember to use the statement.

Variables and Arithmetic Expressions.

- **Variables** in C have the same meaning as variables in algebra. That is, they represent some unknown, or variable, value.

  x = a + b

- Variables in C may be given representations containing multiple characters. But there are rules for these representations.
- Variable names (identifiers) in C
  - May only consist of letters, digits, and underscores
  - May be as long as you like, but only the first 31 characters are significant
  - May not begin with a digit
  - May not be a C **reserved word (keyword)**

Reserved Words (Keywords) in C

auto	break	int	long
case	char	register	return
const	continue	short	signed
Default	do	sizeof	static
double	else	struct	switch
enum	extern	typedef	union
float	for	unsigned	void
goto	if	volatile	while

Naming Conventions.

- C programmers generally agree on the following **conventions** for naming variables.
  - Begin variable names with lowercase letters
  - Use **meaningful** identifiers
  - Separate “words” within identifiers with underscores or mixed upper and lower case.
  - Be consistent!
- C is **case sensitive**.
- Use all uppercase for **symbolic constants** (used in **#define** preprocessor directives).
- Note: symbolic constants are not variables, but make the program easier to read.
- Examples:
  - #define PI 3.14159
  - #define AGE 52

What Does a Variable Have?

Every variable has:

- a name (e.g., number of students),
- an address (i.e., a location in memory, such as 123456),
- a data type (e.g., int, float, char), and
- a value (which may be undefined, also known as garbage). The value is also known as the contents of the variable — that is, the value is the contents of the variable's memory location.

Compile Time and Runtime

- Events that occur while a program is being compiled are said to happen at compile time.
- Events that occur while a program is running are said to happen at runtime.

  For example, the address of a variable is chosen at compile time, while its value often is determined at runtime.

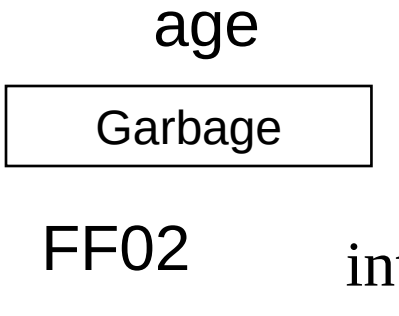
Declaring Variables

- Before using a variable, you must give the compiler some information about the variable; i.e., you must declare it.
- The declaration statement includes the data type of the variable.
- Examples of variable declarations:

```
int age ;
float  area ;
```

- When we declare a variable
  - Space is set aside in memory to hold a value of the specified data type
  - That space is associated with the variable name
  - That space is associated with a unique address
  - Unless we specify otherwise, the space has no known value.
- Visualization of the declaration

```
int  age ;
```



Notes About Variables.

- You must not use a variable until you somehow give it a value.
- You can not assume that the variable will have a value before you give it one.
  - Some compilers do, others do not! This is the source of many errors that are difficult to find.
  - **Assume your compiler does not give it an initial value!**
  - Variables may be given initial values, or **initialized**, when declared.

Displaying Variables.

- Variables hold values that we occasionally want to show the person using the program.
- We have a function called printf( ) that will allow us to do that.
- The function printf needs two pieces of information to display things.
  - How to display it
  - What to display
- printf(“%f\n”, diameter);
- The name of the function is “printf”.

- The %f is known as a placeholder: it holds the place of the value of the variable that we actually want to output.

- Inside the parentheses are:
  - print specification, where we are going to display:
    - a floating point value (“%f”)
    - We want to have the next thing started on a new line (“\n”).
  - We want to display the contents of the variable diameter.

Code	Format
%c	character
%d	signed integers
%i	signed integers
%e	scientific notation, with a lowercase "e"
%E	scientific notation, with a uppercase "E"
%f	floating point
%g	use %e or %f, whichever is shorter
%G	use %E or %f, whichever is shorter
%o	octal
%s	a string of characters
%u	unsigned integer
%x	unsigned hexadecimal, with lowercase letters
%X	unsigned hexadecimal, with uppercase letters
%p	a pointer
%n	the argument shall be a pointer to an integer into which is placed the number of characters written so far
%%	a '%' sign

scanf (“%f”, &diameter);

- The scanf( ) function also needs two items:
  - The input specification “%f”. (Never put a “\n” into the input specification.)
  - The address of where to store the information. (We can input more than one item at a time if we wish, as long as we specify it correctly.)
- Notice the “&” in front of the variable name. It says to use the address of the variable to hold the information that the user enters.
- Comments should explain why you are doing something, not what you are doing it.

Reading Multiple Variables with a Single `scanf`

C allows inputting multiple variables per scanf statement. At runtime, when the user types in the input values, they can separate the individual input values

- by blank spaces, and/or
- by tabs, and/or
- by carriage returns (newlines).
  - Blank spaces, tabs and carriage returns, as a group, are known as white space.

Exercise 01.

- Write a program to read two integers from the stdin and output the sum of two integers them to stdout.

Arithmetic Operators in C

Name	Operator	Example
Addition	+	num1 + num2
Subtraction	-	initial - spent
Multiplication	*	fathoms * 6
Division	/	sum / count
Modulus	%	m % n

- If both operands of a division expression are integers, you will get an integer answer. The fractional portion is thrown away.
- Examples : 17 / 5 = 3
  - 4 / 3 = 1
  - 35 / 9 = 3

Division By Zero.

- Division by zero is mathematically undefined.
- If you allow division by zero in a program, it will cause a **fatal error**. Your program will terminate execution and give an error message.
- **Non-fatal errors** do not cause program termination, just produce incorrect results.

Arithmetic Operators
Rules of Operator Precedence

Operator (s)	Precedence & Associativity
( )	Evaluated first. If **nested (embedded)**, innermost first. If on same level, left to right.
* / %	Evaluated second. If there are several, evaluated left to right.
+ -	Evaluated third. If there are several, evaluated left to right.
=	Evaluated last, right to left.

What is a Constant?

- In mathematics, a constant is a value that cannot change.
- In programming, a constant is like a variable, except that its value cannot change.

There are two categories of constants:

- literal constants, whose values are expressed literally;
- named constants, which have names.

A literal constant is a constant whose value is specified literally:

- int literal constants
  (e.g., 2, 1, 7,98, -1)
- float literal constants
  (e.g., 2.4, 0.6, -14.7)
- char literal constants
  (e.g., 'B', '4', '?')

Named Constants

- A named constant is a constant that has a name.
- A named constant is exactly like a variable, except that its value is set at compile time and CANNOT change at runtime.
- A named constant is exactly like a literal constant, except that it HAS A NAME.
- In a named constant declaration, we indicate that it's a constant via the const attribute, and we MUST initialize it:

```
const float pi = 3.14;
```

- When you embed numeric literal constants in the body of your program, you make it much harder to maintain and upgrade your program.

Repetition and Looping

- Repetition means performing the same set of statements over and over.
- The most common way to perform repetition is via looping.
- A loop is a sequence of statements to be executed, in order, over and over, as long as some condition continues to be true.
- while Loop
  - C has a loop construct known as a while loop:
  - ```
    while (condition) {
        statement1;
        statement2;
        ...
    }
    ```
  - The condition of a while loop is a Boolean expression completely enclosed in parentheses – just like in an if block.
  - The sequence of statements between the while statement's block open and block close is known as the loop body.
    ```

`while` Loop Behavior

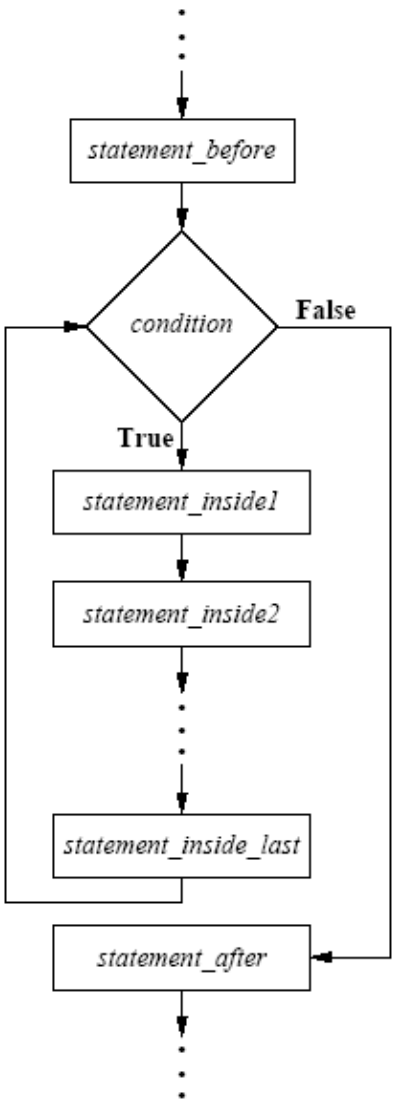
```
while (condition) {
        statement1;
        statement2;
        ...
    }
```

A while loop has to the following behavior:

1. The condition is evaluated, resulting in a value of either true (1) or false (0).
2. If the condition evaluates to false (0), then the statements inside the loop body are skipped, and control is passed to the statement that is immediately after the while loop's block close.
3. If the condition evaluates to true (1), then:
   a. the statements inside the loop body are executed in sequence.
   b. When the while loop's block close is encountered, the program jumps back up to the associated while statement and starts over with Step 1.

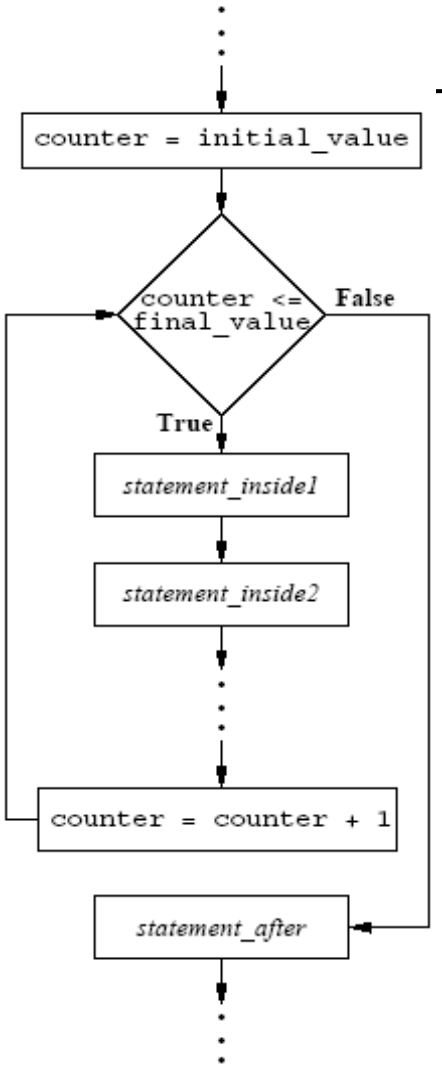
`while` Loop Flowchart

```
statement_before;
while (condition) {
      statement_inside1;
      statement_inside2;
      ...
}
statement_after;
```



for Loop

```
for (counter = initial_value;
      counter <= final value; counter++) {
    statement1;
    statement2;
    ...
} /* for counter */
statement_after;
```



Variables and Arithmetic Expressions.

- **Exercise 01:** Write the program to print the following table of Fahrenheit temperatures and their centigrade or Celsius equivalents.[Use the formula c = 5/9(f – 32).]

0	-17
20	-6
40	4
60	15
.	.

- **Exercise 02:** Modify the program to print the following table.

0	-17.8
20	-6.7
40	4.4
...	