Faculty of Information Technology, University of Moratuwa

BSc. (Hons) in Information Technology

BSc. (Hons) Artificial Intelligence

Programming Fundamentals – IN1101

Level 1 – Semester 1

1. Introduction to C programming

1.1 Program Structure

C program basically consists of the following parts;

- Preprocessor Commands
- Functions
- Variables
- Statements & Expressions
- Comments

```
#include <stdio.h>
int main(void) {
    printf("Welcome to UoM");
}
```

Output: Welcome to UoM

Or

```
#include <stdio.h>
int main() {
    printf("Welcome to UoM");
    return 0;
}
```

Output: Welcome to UoM

Code explanation:

#include <stdio.h> is a directive to the C preprocessor. Lines beginning with # are processed by the preprocessor before compilation. It tells the preprocessor to include the contents of the standard input/output header (<stdio.h>) in the program. This header contains information used by the compiler when compiling calls to standard input/output library functions such as printf.

// indicating that these **two lines are comments**. You insert comments to document programs and improve program readability. Comments do not cause the computer to perform any action when the program is run. Comments are ignored by the C compiler and do not cause any machine language object code to be generated. Comments also help other people read and understand your program.

/*...*/ multi-line comments in which everything from /* on the first line to */ at the end of the last line is a comment.

int main(void) is a part of every C program. The parentheses after main indicate that main is a program building block called a function. C programs contain one or more functions, one of which must be main. Every program in C begins executing at the function main. Functions can return information. The keyword int to the left of main indicates that main "returns" an integer (whole number) value. Simply include the keyword int to the left of main in each of your programs. Functions also can receive information when they're called upon to execute. The void in parentheses here means that main does not receive any information.

A left brace {begins the body of every function. A corresponding right brace ends each function. This pair of braces and the portion of the program between the braces is called a block.

printf("Welcome to UoM"); instructs the computer to perform an action, namely to print on the screen the string of characters marked by the quotation marks. A string is sometimes called a character string, a message or a literal. The entire line, including the printf function (the "f" stands for "formatted"), its argument within the parentheses and the semicolon (;) is called a statement. Every statement must end with a semicolon (also known as the statement terminator). When the preceding printf statement is executed, it prints the message Welcome to UoM on the screen. The characters normally print exactly as they appear between the double quotes in the printf statement.

1.2 Printing multiple statements

```c
#include <stdio.h>
int main(void) {
    printf("Welcome ");
    printf("to UoM");
}
```

Output: Welcome to UoM

```c
#include <stdio.h>
int main(void) {
    printf("Welcome\nto\nUoM\n");
}
```

Output:

```
Welcome
to
UoM
```

Code explanation:

One printf can print several lines by using additional newline characters. Each time the \n (newline) escape sequence is encountered, output continues at the beginning of the next line.

Escape sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop.
\a	Alert. Produces a sound or visible alert without changing the current cursor position.
\\\\	Backslash. Insert a backslash character in a string.
\\"	Double quote. Insert a double-quote character in a string.

1.3 Data Types, User inputs and Variables

Following are examples of some very common data types used in C:

- **char:** The most basic data type in C. It stores a single character and requires a single byte of memory in almost all compilers.
- **int:** As the name suggests, an int variable is used to store an integer.
- **float:** It is used to store decimal numbers (numbers with floating point value) with single precision.
- **double:** It is used to store decimal numbers (numbers with floating point value) with double precision.

Data Type	Format Specifier
int	%d
char	%c
float	%f
double	%lf

Note: %.2f is used to print the fractional value up to two decimal places.

Rules for naming C variables:

- Variable name must begin with letter or underscore.
- Variables are case sensitive.
- They can be constructed with digits, letters.
- No special symbols are allowed other than underscore.

Declaring and initializing variable:

- Variables should be declared in the C program before to use.
- Variable initialization means assigning a value to the variable.

Variable Declaration

A variable must be defined before using it in a program to benefit the compiler. To declare a variable, specify the type of the variable and then its name, followed by the semicolon at the end of the variable declaration statement.

data_type variable_name;

Eg: int height;

If several variables have the same type, their declarations can be combined.

Eg: int height, length, width, volume;

Note: Each complete declaration ends with a semicolon. Declare a variable as close as possible to its first use or at the beginning of the section.

Variable Initialization

Most C compilers allow a variable to be assigned a value when it is declared. This is called initialization. It allows us to declare a variable and assign its value in the same statement.

data_type data_name = expression;

First, the data type is given, followed by the variable name, an equal sign for value assignment. Then an expression or a number is stated whose value is to be assigned to the variable. The expression must be able to be evaluated to a constant value when the program is compiled. If it does not initialize, it contains whatever random junk was in memory.

Eg: int total=0;

```c
#include <stdio.h>
int main(void) {
    int integer1; // first number to be entered by user
    int integer2; // second number to be entered by user
    int sum; // variable in which sum will be stored

    printf("Enter first integer\n"); // prompt
    scanf("%d", &integer1); // read an integer

    printf("Enter second integer\n"); // prompt
    scanf("%d", &integer2); // read an integer

    sum = integer1 + integer2; // assign total to sum

    printf("Sum is %d\n", sum); // print sum
}
```

Output:

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

Code explanation:

The names integer1, integer2 and sum are the names of **variables** locations in memory where values can be stored for use by a program. These definitions specify that variables integer1, integer2 and sum are of type int, meaning they'll hold integer values.

scanf (the “f” stands for “formatted”) uses to obtain a value from the user. The function reads from the standard input, which is usually the keyboard. This scanf has two arguments, "%d" and &integer1. The first, the format control string, indicates the type of data that should be entered by the user. The %d conversion specifier indicates that the data should be an integer (the letter d stands for “decimal integer”). The % in this context is treated by scanf(and printf as we’ll see) as a special character that begins a conversion specifier. The second argument of scanf begins with & called the address operator followed by the variable name. The &, when combined with the variable name, tells scanf the location (or address) in memory at which the variable integer1 is stored. The computer then stores the value that the user enters for integer1 at that location. The use of & is often confusing to novice programmers or to people who have programmed in other languages that do not require this notation. For now, just remember to precede each variable in every call to scanf with &.

sum = integer1 + integer2; calculates the total of variables integer1 and integer2 and assigns the result to variable sum using the assignment operator =. The statement is read as, “sum gets the value of integer1 + integer2.” Most calculations are performed in assignments. The = operator and the + operator are called binary operators because each has two operands. The + operator’s two operands are integer1 and integer2. The = operator’s two operands are sum and the value of the expression integer1 + integer2.

printf("Sum is %d\n", sum); calls function printf to print the literal Sum is followed by the numerical value of variable sum on the screen. This printf has two arguments, "Sum is %d\n" and sum. The first argument is the format control string. It contains some literal characters to be displayed, and it contains the conversion specifier %d indicating that an integer will be printed. The second argument specifies the value to be printed. Notice that the conversion specifier for an integer is the same in both printf and scanf.

Calculations can also be performed inside printf statements. We could have combined the previous two statements into the statement printf("Sum is %d\n", integer1 + integer2);

Local Variables:

- The scope of local variables will be within the function only.
- These variables are declared within the function and can't be accessed outside the function.

Global Variables:

- The scope of global variables will be throughout the program. These variables can be accessed from anywhere in the program.

```c
#include<stdio.h>
int main() {
/* local variable declaration */
    int a, b, c;
    a = 10;
    b = 20;
    c = a + b;
    printf("value of c=%d\n", c);
    return 0;
}
```

Output: value of c= 30

```c
#include<stdio.h>
int c;
int main() {
    int a, b;
    a = 10;
    b = 20;
    c = a + b;
    printf("value of c=%d\n", c);
    return 0;
}
```

Output: value of c= 30

1.4 <u>Constants</u>

Constants refer to fixed values the program may not alter during its execution. Constants are declared in the program in one of the following ways:

- Using the **const** keyword
- Using the **# define** directive, The Syntax:

#define constant_name constant_value const

data_type constant_name=constant_value

```
#include <stdio.h>
#define length 10
int main() {
    int width = 5;
    int area;

    area = length * width;
    printf("value of area %d", area);

    return 0;
}
```

Output: value of area 50

Code explanation:

#define identifier value indicates constant declaration.

1.5 <u>Arithmetic Operators</u>

Operation	**Operator**
Addition	+
Subtraction	-
Multiplication	*
Division	/
Remainder	%

1.6 <u>Assignment Operators</u>

The assignment operator specifies that the first variable_name is assigned the value of the second expression. Expression is evaluated, and the result is assigned to the variable. The expression can be a single variable or a constant. It may contain variables, constants and operators.

The following combined operators are also possible.

x += y;	is the same as x=x+y;
x -= y;	is the same as x=x-y;
x *= y;	is the same as x=x*y;
x %= y;	is the same as x=x%y;

Exercise:

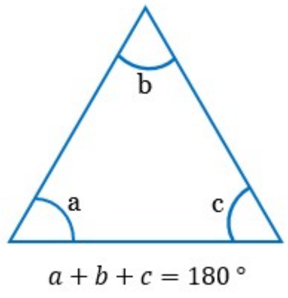
1. Write a program to calculate the area and circumference of a circle.

   Area = PI*radius*radius, circumference = 2*PI*radius

   PI is an arithmetic constant whose approximated value is 3.14159.
2. Write a program to input temperature in Centigrade and convert to Fahrenheit.

   The temperature conversion formula from degree Celsius to Fahrenheit is given by;

   Fahrenheit = (Celsius * 9 / 5) + 32
3. Write a program to the third angle of a triangle if two angles are given. Sum of angles of a triangle is 180 degrees.



If two angles of a triangle are given, then the third angle of the triangle is given by;

C = 180 – (a + b)

4. Write a program to input a number and find the square root of the given number. You may use sqrt() inbuilt function.

   Hint:

   #include <math.h>

   root = sqrt(num);
5. Write a program to input marks of five subjects of a student and calculate the total, average and percentage of all subjects.
6. Write a program to calculate an average fuel consumption from the given total distance (integer value) travelled (in km) and spent fuel (in litres).
7. Write a program to calculate net salary of an employee by reading the values of number of hours and hourly rate through the keyboard.

   Number of hours = 100

   Hourly rate =Rs 250

   Gross salary= Number of hours * Hourly rate
   Deduction rate=8%

   Deduction= Gross salary* Deduction rate

   Net salary = gross salary –deduction