Lab Manual

2

Introduction to Computing

Department of Electrical Engineering
University of Engineering and Technology Lahore 2016
Instructor: Kh. Shahzada Shahid

Name ___________________________
Registration Number ______________

Procedure:

1. Students should read the manual before coming to lab.
2. In the lab, students should complete Labs 2.1 through 2.4 in sequence. Your instructor will give further instructions as to grading and completion of the lab.

Purpose:

1. To understand the Primitive Data Types and Variables in C
2. To learn how to perform Arithmetic Operations and Assignment.

Data Type

Early computer programmers had the burdensome task of having to write their programs in the binary language of the machine they were programming. This meant that computer instructions had to be hand-coded into binary numbers by the programmer before they could be entered into the machine. Furthermore, the programmers had to explicitly assign and reference any storage locations inside the computer's memory by a specific number or memory address. Today's programming languages allow you to concentrate more on solving the particular problem at hand than worrying about specific machine codes or memory locations. They enable you to assign symbolic names, known as variable names, for storing program computations and results.

*A **Variable** name must be chosen by you in a meaningful way to reflect the type of value that is to be stored in that variable.*

Data types in C programming language enables the programmers to appropriately select the data as per requirements of the program and the associated operations of handling it.

Data types in c language can be broadly classified as:
a) Primitive Data Types b) User Defined Data Types c) Derived Data Types.
In this lab we will only focus on primitive data types, user defined and derived data types will be discussed separately.

Primitive Data Types

The primitive data types in c language are the inbuilt data types provided by the c language itself. Thus, all c compilers provide support for these data types. The following primitive data types in c are available:

Integer Data Type, int

Integer data type is used to declare a variable that can store numbers without a decimal. The keyword used to declare a variable of integer type is "int". Following syntax is used to declare and print integer type variable:

int *variable_name*;
int_variable_name = 100;
printf("The value of int_variable_name is %i", *int_variable_name*);

Float data Type, float

Float data type declares a variable that can store numbers containing a decimal number (real numbers). Following syntax is used to declare and print float type variable:

float *float_variable_name*;
float_variable_name = 85.99;
printf("The value of float_variable_name is %f", *float_variable_name*);

Double Data Type, double

Double data type also declares variable that can store floating point numbers but gives precision double than that provided by float data type. Thus, double data type is also referred to as double precision data type. Following syntax is used to declare and print double type variable:

double *double_variable_name*;
double_variable_name = 85.99;
printf("The value of double_variable_name is %f", *double_variable_name*);

Character Data Type, char

Character data type declares a variable that can store a character constant. Thus, the variables declared as char data type can only store one single character. Following syntax is used to declare and print char type variable:

char *char_variable_name*;
char_variable_name = 'A';
printf("The value of char_variable_name is %c", *char_variable_name*);

Data Type Qualifiers

Apart from the primitive data types mentioned above, there are certain data type qualifiers that can be applied to them in order to alter their range and storage space and thus, fit in various situations as per the requirement. Following data type qualifiers are available in c:

a) short b) long c) signed d) unsigned

It should be noted that the above qualifiers cannot be applied to float and can only be applied to integer and character data types.

The entire list of data types in c available for use is given below:

C Data Types		Size(in bytes)	Format Specifier
Integer Data Types	int	2 or 4	%i or %d
	signed int	2 or 4	%i or %d
	unsigned int	2 or 4	%u
	short int	2	%hi
	signed short int	2	%hi
	unsigned short int	2	%hu
	long int	4	%li
	signed long int	4	%li
	unsigned long int	4	%lu
Floating Point Data Types	float	4	%f
	double	8	%g %G or %e %E (for scientific notation)
	long double	10	%Lf %LF, %Lg %LG, %Le %LE
Character Data Types	char	1	%c
	signed char	1	%c
	unsigned char	1	%c

In C, any number, single character, or character string is known as a constant. For example, the number 58 represents a constant integer value. The character string "Programming in C is fun.\n" is an example of a constant character string. Expressions consisting entirely of constant values are called constant expressions. So, the expression

$$128 + 7 - 17$$

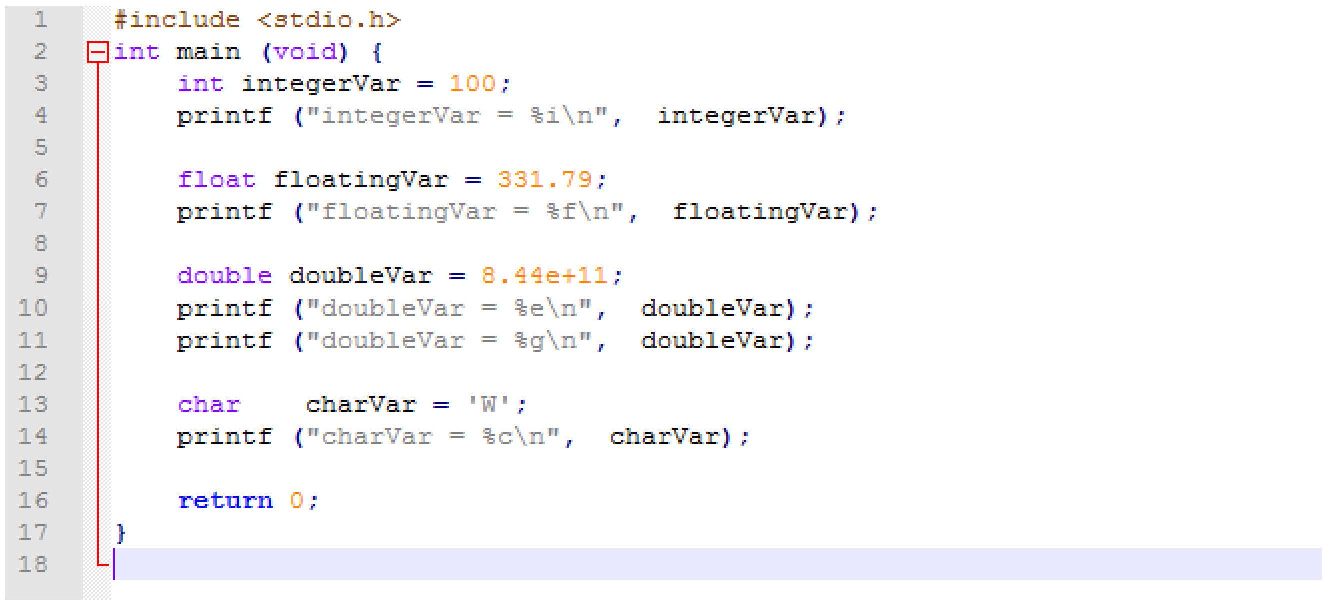
is a constant expression because each term of the above expression is a constant value. But if "i" were declared to be an integer variable, the expression

$$128 + 7 - i$$

would not represent a constant expression.

Lab 2.1:

Type and run the following program in your IDE



```c
#include <stdio.h>
int main (void) {
    int integerVar = 100;
    printf ("integerVar = %i\n",  integerVar);

    float floatingVar = 331.79;
    printf ("floatingVar = %f\n",  floatingVar);

    double doubleVar = 8.44e+11;
    printf ("doubleVar = %e\n",  doubleVar);
    printf ("doubleVar = %g\n",  doubleVar);

    char    charVar = 'W';
    printf ("charVar = %c\n",  charVar);

    return 0;
}
```

Write the Output here:

Output:

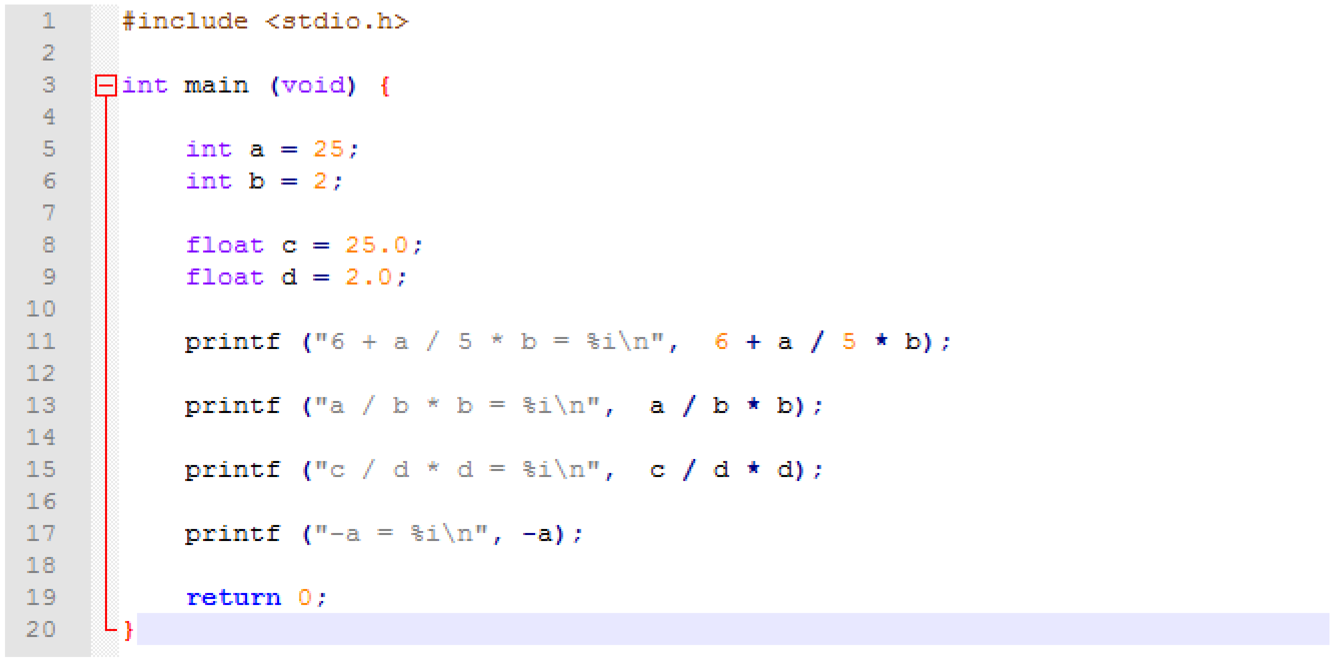
Working with Arithmetic Expressions:

In C, just as in virtually all programming languages, the plus sign (+) is used to add twovalues, the minus sign (–) is used to subtract two values, the asterisk (*) is used to multiplytwo values, and the slash (/) is used to divide two values. These operators are knownas binary arithmetic operators because they operate on two values or terms. Program 2.2 further illustrates the operations of subtraction, multiplication, and division. Each operator in C has a precedence associated with it. This precedence is used to determine how an expression that has more than one operator is evaluated: The operator with the higher precedence is evaluated first. Expressions containing operators of the same precedenceare evaluated either from left to right or from right to left, depending on the operator. This is known as the associative property of an operator.

Arithmetic expressions are evaluated using DMAS rule, i.e. Division, Multiplication, Addition, and Subtraction

Lab 2.2

Analyze the following code



```c
#include <stdio.h>

int main (void) {

    int a = 25;
    int b = 2;

    float c = 25.0;
    float d = 2.0;

    printf ("6 + a / 5 * b = %i\n",  6 + a / 5 * b);

    printf ("a / b * b = %i\n",  a / b * b);

    printf ("c / d * d = %i\n",  c / d * d);

    printf ("-a = %i\n", -a);

    return 0;
}
```

Write the output of the above program without running it

Output:

Now compose and run the program 2.2 in IDE and write the output

Output:

Lab 2.3

Write a C program to print the area and volume of sphere with proper message Formulas are:
Area =4*PI*R*R ;Volume = 4/3*PI*R*R*R where variables PI = 3.14 and R= 5

Output:

Lab 2.4

Write a C program to convert **Centigrade into Fahrenheit**and print the temperature with the proper message.
Formula: **`C= (F-32)/1.8`** **`where c = 54`**

Output: