**Tutorial - Week 1**

# Objectives:

To discuss basic elements of C such as

* general structure of a C program
* variable declarations and data types
* input / output
* arithmetic expressions

1. **Which of the following definitions are correct? If incorrect, give the reason:**
2. int numStudents = 370;
3. int numLabs = 45.5;
4. double 1value, 2value;
5. char grade = 'P';
6. char stLevel = A;
7. char pinNum = 117;
8. number = 24;
9. #define PI = 3.1416;
10. **correct**
11. Syntactically, it is correct. Logically,  **it might be incorrect as you are assigning a real number to an integer variable, the fractional part will be removed but you won’t get any error message**
12. **incorrect** (variable identifier (2value) starts with a number)
13. **correct**
14. **incorrect** (the character is missing the single quotation, unless A has already been declared as a char
15. **correct**
16. **incorrect** (no data type)
17. **incorrect** (there should be no equal sign and semi colon at the end)
18. **Assuming the following variable declarations, what output is produced by printf()?**

int numOfModules = 15, moduleCode = 7;

float width = 65.243, height=38.159;

char secLevel = 'B';

1. printf("Number of Modules=%6d\n", numOfModules);

Displayed Output:

Number of Modules= ~~~~**15**

1. printf("Module Code = %04d\n", moduleCode);

Displayed Output:

Module Code = **0007**

1. printf("Width = %-8.2fHeight = %-8.2f\n", width, height);

Displayed Output:

**Width = 65.24**~~~**Height** = **38.16**~~~

1. printf("Dimensions = %.1f x %.1f mm\n", width, height);

Displayed Output:

**Dimensions = 65.2 x 38.2 mm**

1. printf("Security Level:%3c\n", secLevel);

Displayed Output:

~~**B**

1. **Is scanf() used correctly?**

float x, y;

1. scanf(“Enter a number: %f”, &x);
2. scanf( “%d” , &x);
3. scanf( “%3f”, &x);
4. scanf( “%f, %f”, &x, &y);

**Please note that the above statements will compile. However, an incorrect scanf text format leads to incorrect read operation and hence variable value.**

1. Incorrect use: quoted text in scanf first parameter is not expected.
2. Incorrect as x is a float, not an integer
3. Only 3 digits, including the decimal point, will be read from the user’s input.
4. Incorrect. No comma should be used between the format specifiers in scanf.
5. **Correct mistakes in the program that converts a temperature in Fahrenheit to Celsius: Celsius = 0.55\*(fahrenheit – 32)**

#include <stdio.h>

#define COEFF = 0.55

int main(void)

{

float celsTemp, fahrTemp

printf(' Enter the temperature in Fahrenheits: ');

scanf("%f", fahrTemp );

celsTemp = COEFF \* fahrTemp – 32;

printf("Celsius temperature = %.1f" , fahrtemp );

return 0;

}

* #define COEFF = 0.55 should be #define COEFF 0.55 since COEFF is a symbol name
* float celsTemp, fahrTemp should be float celsTemp, fahrTemp;
* printf(' Enter the temperature in Fahrenheits: '); should be

printf(“ Enter the temperature in Fahrenheits:”);

* scanf("%f", fahrTemp ); should be scanf("%f", &fahrTemp );
* celsTemp = COEFF \* fahrTemp – 32; should be

celsTemp = COEFF \*(fahrTemp – 32);

* printf("Celsius temperature = %.1f" , fahrtemp ); should be printf("Celsius temperature = %.1f" , celsTemp );

1. **Which of the following declarations are correct? If incorrect, give the reason:**
   * + 1. char productType = 'V';
       2. char minutes = 45;
       3. char days =172;
       4. char value = -15;
       5. float width = 32.157e2;
       6. int length = width;

Syntactically, the above statements are correct. Logically, the statement

1. is correct
2. is correct
3. leads to an overflow as char values range is between -128 and 127
4. is correct
5. is correct
6. demotes the width value from float to integer, i.e. its fractional part will be truncated.
7. **What value will be assigned to the variables (taking into account all the preceding operations)**

int intRes, number = 5;

float fpRes;

char grade = 'A';

* + - 1. intRes = 10/3;
      2. intRes = 10%3;
      3. intRes = 15/2/3;
      4. fpRes = 15.0/2;
      5. intRes = number++;
      6. intRes = ++number;
      7. intRes = (7 + 3)/2;
      8. intRes += 4;
      9. fpRes = intRes/2;
      10. fpRes = (float)intRes/2;
      11. grade += 2;

1. 3 : result of integer division as both operands are integers
2. 1: remainder of the division of the operands
3. 2 : parenthesized version of the expression is ( (15/2) /3)= (7/3)=2
4. 7.5 : real division since one operand is not a whole number
5. 5 : post-incrementation operator, the current value of number is first assigned to intRes before being incremented. The resulting values of intRes and number are 5 and 6, respectively.
6. 7 : pre-incrementation operator, the variable number is first incremented before being assigned to intRes. The resulting value of ntRes and number 7 and 7.
7. 5 as the integer division of 10 by 2
8. 9 : compound operator “+=”, the lvalue (intRes) is incremented by the rvalue (4)
9. 4.0 : the integer division is promoted to float. The magnitude remains the same.
10. 4.5 : As we are casting the nominator to the float, the division execution will be real division and no longer integer
11. 'C' : use of the compound operator “+=”; the addition is on a character variable; an increment by 2 on an alphabet letter returns the alphabet letter at offset 2 from the current one
12. **Evaluate the following:**

**float fa = 29.0, fb = 10.0, fc = 10.37;**

**int ia;**

1. ia = (int)(fa/fb);
2. ia = fa/fb;
3. How to round a float to the nearest int?
4. How to assign only the integer part of fc to fa?
5. ia = 2; casting the real division result (2.9) to integer
6. ia = 2; converting the real division result(2.9) to integer (removing the fractional part)
7. (int) (fa + 0.5) if fa is positive and (int) (fa - 0.5) if it is negative. Otherwise, use the function round()
8. fa = (int) fc;
9. **Specify the** lvalue **and** rvalue **in the following statements?**

int a, b=1;

1. a = 1;
2. ++a;
3. b = b + a;
4. a = b++;
5. b = 10++;
6. b = ++(1-a);
7. The value *1* is a rvalue, whereas the variable *a* is a lvalue
8. The variable *a* is a lvalue
9. The expression b + a is a rvalue, whereas *b* is an lvalue
10. a copy of *b* becomes a rvalue and assigned to the lvalue *a*, then the lvalue b is incremented
11. Error: the post incrementation/decrementation can only be applied on a variable, 10 is not a variable
12. Error: the post incrementation/decrementation can only be applied on a variable, (1-a) is an expression and not a variable
13. **Write the #define preprocessor directive and declarations for a program that has a constant macro for PI (3.14159) and variables radius, area, and circumf declared as double, variable num\_circ as an int, and variable circ\_name as a char.**

#include <stdio.h>

#define PI 3.14159

int

main(void)

{

double radius, area, circumf;

int num\_circ;

char circ\_name;

/\* executable statements omitted \*/

}

1. **Write a statement that displays the following line with the value of the type int variable n before the period.**

The value of n is \_\_\_\_\_\_\_\_\_\_\_.

printf("The value of n is %d.\n", n);

1. **Assuming that side and area are type double variables containing the length of one side in cm and the area of a square in square cm, write a statement that will display this information in this form:**

The area of a square whose side length is \_\_\_\_\_\_ cm is \_\_\_\_\_\_ square cm.

printf("The area of a square whose side length is %lf cm is %lf square

cm.\n", side, area);

The l modifier is required in scanf with double, but not in printf.

1. **Show how the value −3.6175 would be printed using the formats %8.4f, %8.3f, %8.2f, %8.1f, %8.0f, %.2f.**

value = -3.6175 (# means blank)

Format Output

%8.4f #-3.6175

%8.3f ##-3.618

%8.2f ###-3.62

%8.1f ####-3.6

%8.0f #####-4.

%.2f -3.62