TUTORIAL 1 - 5

M.K.E.A.D.JAYASINGHE

INDEX NO : 30362

23.1 BATCH

SOFTWARE ENGINEERING (PLYMOUTH)

Tutorial 01

1. People use computer to accomplish various tasks. But the only language the computers can understand is Machine Language. So people are using programming languages in order to interact with the computers. Programming language is the language which facilitate the communication between human and the computer.

2. a.

Differences	
Source Code	**Machine Code**
• High Level Language	• Low Level Language
• Generated by human	• Generated by machine
• Easy to modify	• Difficult to modify
Similarities	
• The instructions are same in both codes despite the language they are written in.	

b.

Differences	
High Level Language	**Low Level Language**
• Easy to understand and learn	• Difficult to understand and learn
• Slow in execution	• Fast in execution
• Uses English like language	• Uses binary code
• Machine independent	• Machine-dependent
• Portable	• Not portable
Similarities	
• Need a translator to convert the code in to executable code.	
• Both are used by programmers.	

c.

Differences	
Compiler	**Interpreter**
• Translates the entire code at once	• Translates the code line by line
• Intermediate object code is generated	• No intermediate object code is generated
• More memory requires	• Less memory requires
Similarities	
• Converts High Level Language code into machine code.	

d.

Differences	
Structured Language	**Object Oriented Language**
• Data and functions are treated separately.	• Data and functions are combined into objects.
• Breaks program into small functions.	• Represents real world entities.
Similarities	
• Both aim to make code easy to read, understand and maintain.	
• Promotes code reusability.	

e.

Differences	
C	**C++**
• Does not support object oriented programming.	• Supports object oriented programming.
• No automatic memory management.	• Having an automated memory management.
Similarities	
• Bothe shares a similar syntax.	
• Both languages should be compiled.	

f.

Differences	
C++	**Java**
• Memory management is done manually.	• Having an automated memory management.
• Supports both procedural & object oriented programming.	• Primarily focuses on object oriented programming.
Similarities	
• Both shares a similar syntax.	
• Supports object oriented programming.	

g.

Differences	
Syntax Error	**Logical Error**
• Occur due to violations of language's grammar (syntax).	• Occur when the code doesn't produce expected outcome.
• Errors will prevent the code from compiling or executing.	• Code may compile and execute without an error.

Similarities
• Both are errors that can be occurred in programming.
• Both can affect to the execution of the program.
• Both errors can be prevented through proper programming practices.

Tutorial 02

1) Single line comments : //the comment…
   Multi line comments : /*the comment…*/

   The purpose of a comment is to explain the functionality of the code. This makes programmer easy to understand, maintain, and debug the program.

2) main()

3) The purpose of **scanf** is to read input from the user.

4) Yes, standard C is a **Case sensitive** language.

5) a) Valid
   b) Invalid : It starts with a digit, which is not allowed.
   c) Invalid : Hyphen is not allowed in identifiers.
   d) Valid
   e) Valid
   f) Valid
   g) Invalid : Spaces are not allowed in identifiers.
   h) Invalid : Hyphen is not allowed in identifiers.
   i) Valid
   j) Invalid : Hyphen is not allowed in identifiers as well as an identifier cannot be started with a digit.

6)
   a) False - printf does not always print in a new line
   b) False – Comments doesn't display in the output
   c) True
   d) True
   e) True
   f) False - C is a case sensitive language
   g) False – Single printf statement can be used to display multiple lines

7)
```
*
**
***
****
*****
```

8) a) scanf("d",&value);
b) printf("The product of %d and %d is %d\n", x, y);
c) scanf("%d",&anInteger);
d) printf("Remainder of %d divided by %d is x % y \n", x, y);
e) printf("The sum is %d\n", x + y);
f) printf("The value you entered is: %d\n",value);

9) a) 2
b) 4
c) x=
d) x=2
e) 5=5
f) Nothing
g) Nothing
h) Nothing
i) Breaks into a new line, but print nothing.

10) a) True
b) True
c) False : **printf** is not an assignment statement.
d) False : C follows the rules of operator precedence while doing arithmetic operations.
e) False : h22 is a valid variable name.

Tutorial 03

1. i) x=x+1;
ii) x+=1;
iii) x++;
iv) ++x;

2. a) z = x++ +y ;
b) product * = 2 ;
c) product = product * 2 ;
d) if (count >10) printf (“Count is greater than 10.”) ;
e) total - = - -x ;
f) total + = x + + ;
g) i) q = q % divisor ;
ii) q % = divisor ;
h) printf (“ %.2f ”, 123.4567) ; - The value that will be printed is 123.46
i) printf (“ %3.f ”, 3.14159) ; - The value that will be printed is 3.142

3. a) scanf(“%d” ,&x) ;
b) scanf(“%d” ,&y) ;
c) int I = 1 ;
d) int power = 1 ;

e) power * = x ;
f) i + + ;
g) while (i <= y)
h) printf (“%d” ,power) ;

Tutorial 04

1. i) The condition should be in **parentheses ()** in the **if** statement.
   ii) When assigning the value 4 to **numNeighbors**, the == should be used instead of single equal sign.
   iii) The code should be written inside the **curly brackets {}**.

2. Process
   Integer type variable called **number** is assigned the value **4** and double type variable called **alpha** is assigned the value **-1.0**.
   Then checks if the value of **number** is greater than zero and if it is true, again it checks if the value of **alpha** is greater than zero. If it is true, then it prints **Here I am!** And if the value of **alpha** is not greater than zero, it prints **No, I’m here!**.
   If he value of **number** is not greater than zero, it prints **No, actually, I’m here!**.

   Output
   As the values assigned to the variables initially, **if(number>0)** condition becomes true.
   Then it checks the **if(alpha>0)** condition and as -1.0 is less than 0, it becomes false.
   So it prints **No, I’m here!**
   And then prints **No, actually, I’m here!** In a new line.
   Then the program will terminate.

   No, I’m here!

   No, actually, I’m here!

3. Truth Table for the given scenario :

doesSignificantWork	makesBreakthrough	nobelPrizeCandidate
False	False	False
False	True	False
True	False	False
True	True	True

So the final value of **nobelPrizeCandidate** depends on the values of both **doesSignificantWork** and **makesBreakthrough.**

The **nobelPrizeCandidate** becomes true only if both **doesSignificantWork** and **makesBreakthrough** get the Boolean value True. If any of them got False, **nobelPrizeCandidate** becomes False.

4. i)

```
{char taxCode;
float taxRate,price;
if (taxCode == 'T')
{
    price += (taxRate / 100) * price;
}}
```

ii)

```
{int opCode;
double X,Y;
if (opCode == 1)
{
   scanf("%lf %lf", &X, &Y);
   sum = X + Y;
   printf("The sum of X and Y is: %lf\n", sum);
}}
```

iii)

```
{int currentNumber;
if (currentNumber % 2 == 1)
 {
   currentNumber = (3 * currentNumber) + 1;
}
Else
 {
   currentNumber = currentNumber / 2;
}}
```

iv)

```
{int year,leapYear;
if (year % 4 == 0)
  {  if (year % 100 == 0)
      { if (year % 400 == 0)
          {leapYear = true;}
        else
          {leapYear = false; }
      } else
        {leapYear = true;}
} else
   {leapYear = false;}}
```

v)

```
{double cost;
int distance;
if (distance >= 0)
  { if (distance <= 100)
     {cost = 5.00;}
   else
     { if (distance <= 500)
        {cost = 8.00;}
      else
        { if (distance < 1000)
           {cost = 10.00;}
         else
           { cost = 12.00;}
        }
     }
  }
}}
```

Tutorial 05

1. Switch

```
{float n1,n2;
int op;
printf("Enter 02 numbers : ");
scanf("%f %f",&n1,&n2);
printf("1. +\n2. -\n3. *\n4. /\n");
printf("Please enter your choice : ");
scanf("%d",&op);
switch(op)
{
   case 1:printf("Addition is %.2f",n1+n2);break;
   case 2:printf("Subtraction is %.2f",n1-n2);break;
   case 3:printf("Multiplication is %.2f",n1*n2);break;
   case 4:printf("Division is %.2f",n1/n2);break;
}}
```

2. While loop

```
1.  { int n,odd=0,even=0,c=1;
     while(c<=10)
    {
        printf("Enter number %d : ",c);
        scanf("%d",&n);
```

```
    if(n%2==0)
        even+=1;
    else
       odd+=1;
    c++;
}
printf("\nThere are %d even numbers\n",even);
printf("There are %d odd numbers\n",odd);}
```

2.
```
{ int n,odd=0,even=0,c=1;
 while(n!=-99)
 {
    printf("Enter number %d : ",c);
    scanf("%d",&n);
    if(n%2==0)
        even+=1;
    else
        odd+=1;
    c++;
 }
 printf("\nThere are %d even numbers\n",even);
 printf("There are %d odd numbers\n",odd);}
```

3. Do while loop

1.
```
{int n,odd=0,even=0,c=1;
 do
 {
    printf("Enter number %d : ",c);
    scanf("%d",&n);
    if(n%2==0)
       even+=1;
    else
       odd+=1;
       c++;
 }
 while(c<=10);
 printf("\nThere are %d even numbers\n",even);
 printf("There are %d odd numbers\n",odd);}
```

2. .
```
{ int n,odd=0,even=0,c=1;
 do
 {
    printf("Enter number %d : ",c);
    scanf("%d",&n);
    if(n%2==0)
```

```
            even+=1;
        else
            odd+=1;
        c++;
    }
    while(n!=-99);
    printf("\nThere are %d even numbers\n",even);
    printf("There are %d odd numbers\n",odd);}
```

4. For loop

1.
```
{ int c;
 float num,sum=0,avg;
 for(c=1;c<=10;c++)
{
  printf("Enter number %d : ",c);
  scanf("%f",&num);
  sum+=num;
}
avg=sum/10;
printf("Average is %.2f ",avg);}
```

2.
```
{ int a,b;
 for(a=1;a<=5;a++)
{
  for(b=1;b<=a;b++)
    {
      printf("* ");
    }
  printf("\n");
}}
```