Yapa Y M D H

30459

23.1

Tutorial 01.

1) Briefly explain the need of a programming language?
    - Programming language directs a computer to complete these commands over and over again so people do not have to do the task repeatedly.

2) Compare and contrast the differences between followings;
    a. Source Code vs. Machine Code
    - The source code is programming language specific code which is non-executable but standed to be converted where as the machine code is the actual executable code

    b. High Level Language vs. Low Level Language
    - High Level Languages requires the use of a compiler or an interpreter for there translation into the machine code.
    - Low Level Languages requires an assembler for directly translating the instruction of the machine language

    c. Compiler vs. Interpreter
        - A compiler takes a program as a whole an interpreter takes single line of of a code

    d. Structured Language vs. Object Oriented Language
        - The structured programming allows developing a program using a set of modules or functions while the object oriented programming allows constructing a program using a set of objects and their intractions

    e. C vs. C++
        - C and C++ have similarities, C++ expands on C by adding OOP features and providing a more extensive standard library. The choice between the two depends on the project requirements and the developer's familiarity with the languages.

    f. C++ v. Java

    - C++ and Java depends on factors such as project requirements, desired performance, platform compatibility, and individual preference. C++ is well-suited for performance-critical applications, while Java excels in cross-platform development and enterprise environments.

g. Syntax error vs. Logical error

- Syntax Error:

A syntax error occurs when the code violates the rules of the programming language's syntax. It means that the code is written in a way that the compiler or interpreter cannot understand. Syntax errors prevent the code from being executed or compiled successfully. Common examples of syntax errors include missing or mismatched punctuation, incorrect variable or function names, and incorrect use of language keywords.

- Logical Error:

A logical error, also known as a semantic error, occurs when the code does not function as intended or produces incorrect results. It is a mistake in the logic or algorithm of the program. Unlike syntax errors, logical errors do not cause the code to fail to compile or execute. Instead, they result in undesired outputs or unexpected behavior during runtime. Logical errors can be challenging to identify and fix as they require careful analysis and debugging of the code's logic.

In summary, a syntax error is a violation of the language's rules, preventing successful compilation or execution, while a logical error is a mistake in the program's logic, resulting in incorrect outputs or behavior during runtime.

Tutorial 02.

1. How do you write comments in a c program? What is the purpose of comments in a program?
    - Single line comments starts with two forword slashes (//)
    - Commenting involves placing human readable descriptions inside of computer program detailing what code is doing.

2. Which is the function that is essential in a C program?
    - The “main” function
3. What is the purpose of ‘scanf’ ?
    - It allows you to read input from the user or form a file and store that input in variables of different data types.
4. Is ‘standard c’ a case sensitive language?
    - Yes,the C language is case sensitive

5. Determine which of the following are valid identifiers. If invalid, explain why.

(a) record1

Valid identifier.

(b) 1record

Invalid identifier. It starts with a numeric digit, which is not allowed as the first character of an identifier in C. Identifiers must start with a letter or an underscore.

(c) file-3

Invalid identifier. It contains a hyphen, which is not allowed in C identifiers. Only letters, digits, and underscores are permitted.

(d) return

Valid identifier. Although "return" is a keyword in C, it is allowed as an identifier.

(e) $tax

Valid identifier. It begins with a dollar sign, which is allowed in C identifiers. However, using a dollar sign as a prefix is not a common convention in C programming.

(f) name

Valid identifier

(g) name and address

Invalid identifier. It contains spaces, and spaces are not allowed in C identifiers. Instead, you can use underscores to separate words, like "name_and_address".

(h) name-and-address

Invalid identifier. It contains a hyphen, which is not allowed in C identifiers.

(j) 123 - 45 - 6789

Invalid identifier. It starts with a numeric digit and contains hyphens, which are not allowed in C identifiers.

6. State whether each of the following is true or false. If false, explain why.

a) Function printf always begins printing at the beginning of a new line.

- False. The `printf` function in C does not automatically begin printing at the beginning of a new line. It continues printing from the current cursor position on the same line unless specified otherwise.

b) Comments cause the computer to print the text enclosed between /* and */ on the screen when the program is executed.

- False. Comments in C, denoted by `/*` and `*/`, are ignored by the compiler and do not affect the execution of the program. They are used for providing explanatory notes or disabling code temporarily. The text enclosed between `/*` and `*/` is not printed on the screen when the program is executed.

c) The escape sequence \n when used in a printf format control string causes the cursor to position to the beginning of the next line on the screen.

- True. The escape sequence `\n` is used in C's `printf` function to insert a newline character. When used within a `printf` format control string, `\n` causes the cursor to position at the beginning of the next line on the screen.

d) All variables must be defined before they're used.

- False. In C, it is not required to define all variables before they are used. However, variables should be declared before they are used to inform the compiler about their existence and type. The actual definition (assignment of initial value) can occur later in the code.

e) All variables must be given a type when they're defined.

- True. In C, all variables must be given a type when they are defined. The type defines the nature of the variable, such as integer, floating-point, character, etc., and determines the memory allocated to it and the operations that can be performed on it.

f) C considers the variables, number and NuMbEr to be identical.

- False. C is case sensitive, so the variables `number` and `NuMbEr` would be considered as distinct and separate variables with different names.

g) A program that prints three lines of output must contain three printf statements.

- False. A program that prints three lines of output does not necessarily require three `printf` statements. It could use a single `printf` statement with appropriate formatting, including escape sequences like `\n` to introduce line breaks within the output. Multiple lines of output can be generated with a single `printf` statement

7. What does the following code print? printf("*\n**\n***\n****\n*****\n")

```
*

**

***

****

*****
```

8. Identify and correct the errors in each of the following statements. (Note: There may be more than one error per statement.)

a) scanf("d", value);

- Error:= The format specifier in `scanf` is missing the `%` symbol before the `d`.
- Correction:=`scanf("%d", &value);`

b) printf("The product of %d and %d is %d"\n, x, y);

- Errors:= Missing a comma at the end of the format string. The closing double quote should be placed after the comma
- Correction:=`printf("The product of %d and %d is %d\n", x, y);`

c) Scanf("%d", anInteger);

- Error: = The function `Scanf` is written with an uppercase 'S'. C is a case-sensitive language, so the correct function name should be `scanf`.
- Correction:= `scanf("%d", &anInteger);`

d) printf("Remainder of %d divided by %d is\n", x, y, x % y);

- No errors

e) print("The sum is %d\n," x + y);

- Error:= The function `print` is undefined in C. The correct function for printing is `printf`. The comma is missing between the format string and the argument list.
- Correction:= `printf("The sum is %d\n", x + y);`
-

f) Printf("The value you entered is: %d\n, &value);

- Error:= The function `Printf` is written with an uppercase 'P'. C is a case-sensitive language, so the correct function name should be `printf`. The ampersand symbol should not be included before `value` since it's not necessary for `printf`.

- Correction: = `printf("The value you entered is: %d\n", value);`

9. What, if anything, prints when each of the following statements is performed? If nothing prints, then answer "Nothing." Assume x = 2 and y = 3 .

a) printf("%d", x); = "2".

b) printf("%d", x + x); ="4".

c) printf("x="); = "x="

d) printf("x=%d", x); = "x=2"

e) printf("%d = %d", x + y, y + x); "5=5"

f) z = x + y; =Not printed

g) scanf("%d%d", &x, &y); = Not printed

h) /* printf("x + y = %d", x + y); */= Not printed

i) printf("\n"); = Not printed

10. State which of the following are true and which are false. If false, explain your answer.

a) C operators are evaluated from left to right.

- True

b) The following are all valid variable names: _under_bar_ , m928134 , t5 , j7 , her_sales , his_account_total , a , b , c , z , z2 .

- True

c) The statement printf("a = 5;"); is a typical example of an assignment statement.

- False

d) A valid arithmetic expression containing no parentheses is evaluated from left to right.

- True

e) The following are all invalid variable names: 3g , 87 , 67h2 , h22 , 2h

- True

Tutorial 03.

Q1. Write four different C statements that each add 1 to integer variable x.

- x=x+1
- x+=1
- x++
- ++x

Q2. Write a single C statement to accomplish each of the following:

a) Assign the sum of x and y to z and increment the value of x by 1 after the calculation.
   - z = x++ + y

b) Multiply the variable product by 2 using the *= operator.
   - product *= 2;`

c) Multiply the variable product by 2 using the = and * operators.
   - product = product * 2;

d) Test if the value of the variable count is greater than 10. If it is, print "Count is greater than 10."

```
if (count > 10)
{
printf("Count is greater than 10.");
}
```

e) Decrement the variable x by 1, then subtract it from the variable total.

   - total -= --x;

f) Add the variable x to the variable total, then decrement x by 1.

- total += x-- - 1;

g) Calculate the remainder after q is divided by divisor and assign the result to q. Write this statement two different ways.

h) Print the value 123.4567 with 2 digits of precision. What value is printed?
   - printf("%.2f", 123.4567);

i) Print the floating-point value 3.14159 with three digits to the right of the decimal point. What value is printed?
   - printf("%.3f", 3.14159);

Q3. Write single C statements that

a) Input integer variable x with scanf.
   - scanf("%d", &x);

b) Input integer variable y with scanf.
   - scanf("%d", &y);

c) Initialize integer variable i to 1.
   - int i = 1;

d) Initialize integer variable power to 1.
   - int power = 1;

e) Multiply variable power by x and assign the result to power.
   - power *= x;

f) Increment variable i by 1.
   - i++;

g) Test i to see if it's less than or equal to y in the condition of a while statement.
   - while (i <= y)

h) Output integer variable power with printf
   - printf("%d", power);

Tutorial 04.

01) What is wrong with the following if statement (there are at least 3 errors). The Indentation indicates the desired behavior.

```
 if numNeighbors >= 3 || numNeighbors = 4
++numNeighbors;
printf("You are dead! \n " );
else
--numNeighbors;
```

i. (numNeighbors == 4)

ii.(numNeighbors >= 3) || (numNeighbors == 4)

iii.
```
        {
                ++numNeighbors;
                 printf("You are dead! \n");
        }
        else
        {
                  --numNeighbors;
        }
```

02) Describe the output produced by this poorly indented program segment:

```
int number = 4;

double alpha = -1.0;

if (number > 0)

if (alpha > 0)

printf("Here I am! \n" );

else

printf("No, I’m here! \n");

printf(“No, actually, I’m here! \n");
```

The "No, actually, I'm here!" message is a printed.

03) Consider the following if statement, where doesSignificantWork, makesBreakthrough, and nobelPrizeCandidate are all boolean variables:

```
 if (doesSignificantWork)
{
          if (makesBreakthrough)
          nobelPrizeCandidate = true;
          else
          nobelPrizeCandidate = false;
 }
else if (!doesSignificantWork) nobelPrizeCandidate = false;
```

04)Write if statements to do the following

– If character variable taxCode is ’T’, increase price by adding the taxRate percentage of price to it

– If integer variable opCode has the value 1, read in double values for X and Y and calculate and print their sum

– If integer variable currentNumber is odd, change its value so that it is now 3 times currentNumber plus 1, otherwise change its value so that it is now half of currentNumber (rounded down when currentNumber is odd).

– Assign true to the boolean variable leapYear if the integer variable year is a leap year. (A leap year is a multiple of 4, and if it is a multiple of 100, it must also be a multiple of 400.)

– Assign a value to double variable cost depending on the value of integer variable distance as follows:

Distance	Cost
0 through 100	5.00
More than 100 but not more than 500	8.00
More than 500 but less than 1,000	10.00
1,000 or more	12.0

Tutorial 05.

Switch

Input two numbers and display the outputs of the basic mathematic operations. The output screen should be displayed as follows;

```
Enter two numbers ____  ____

1. +
2. –
3. *
4. /

Please enter your Choice ___
```

```
 #include <stdio.h>
#include <stdlib.h>
 int main()
     {
      int n1,n2,choice,sum,sub,mul,div;
                    printf("Enter two numbers: ");
                   scanf("%d",&n1);
 scanf("%d",&n2);
 printf("1.+ \n2.- \n3.* \n4./ \n");
```

```
  printf("Enter your choice: ");
 scanf("%d",&choice);
 sum=n1+n2;
 sub=n1-n2;
 mul=n1*n2;
 div=n1/n2
switch (choice)
         {
            case 1:printf("sum = %d",sum);break;
            case 2:printf("substraction = %d",sub);break;
            case 3:printf("multiplication = %d",mul);break;
            case 4:printf("division= %d",div);break;
       }
          return 0;
}
```

While loop

1. Input 10 numbers and display the total count of odd & even numbers in the entered number series.

```
#include <stdio.h>
 #include <stdlib.h>
int main()
 {
int counter=1,odd=0,even=0,no;
while (counter<=10)
{ printf("Enter %d number: ",counter);
scanf("%d",&no);
```

```
  if (no%2==0)
  { even=even+1; }
 else
 { odd=odd+1; }
 counter=counter+1;
 }
 printf("Total count of odd numbers: %d\n",odd);
  printf("Total count of even numbers: %d\n",even);
 return 0;
 }
```

2. Modify the above program in to enter series of numbers terminates when the user enter -99 and display the same expected output

```
 #include <stdio.h>
#include <stdlib.h>
 int main()
 {
int counter=1,odd=0,even=0,no;
while (counter<=10,no!=-99)
{
          printf("Enter %d number: ",counter);
         scanf("%d",&no);
             if (no%2==0)
             { even=even+1; }
        else
        { odd=odd+1; }
          counter=counter+1;
     }
     printf("Total count of odd numbers: %d \n",odd);
    printf("Total count of even numbers: %d \n",even);
```

```
        return 0;
        }
```

Do while loop

Rewrite the programs for the above while loop question 1 & 2 using do while loop

Q1

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int counter=1,odd=0,even=0,no;
 do
{
printf("Enter %d number: ",counter);
scanf("%d",&no);
 if (no%2==0)
 { even=even+1; }
Else
 { odd=odd+1; }
counter=counter+1;
}
while (counter<=10);
printf("Total count of odd numbers: %d \n",odd);
printf("Total count of even numbers: %d \n",even);
return 0;
}
Q2
 #include <stdio.h>
#include <stdlib.h>
int main()
{
int counter=1,odd=0,even=0,no;
do
{
printf("Enter %d number: ",counter);
 scanf("%d",&no);
if (no%2==0)
 { even=even+1; }
else
 { odd=odd+1; }
counter=counter+1;
}
while (counter<=10,no!=-99);
 printf("Total count of odd numbers: %d \n",odd);
printf("Total count of even numbers: %d \n",even);
return 0;
}
```

For loop

1. Input 10 numbers and display the average value using the for loop

   ```
   #include <stdio.h>
    #include <stdlib.h>
    int main()
   {
   ```

```
int no,sum=0,i;
float avg;
for (i=1;i<=10;i++)
{
printf("Enter the number: ");
 scanf("%d",&no);
sum=sum+no;
}
avg=sum/10;
 printf("Sum is %d \n",sum);
printf("Average is %.2f ",avg);
return 0;
}
```

2. Display the following output using the for loop

```
*
**
***
****
*****
```

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int i,x,row=5;
for (i=1;i<=row;i++)
{
for (x=1;x<=i;++x)
 {
```

```
printf("*");
}
printf("\n");
}
return 0;
}
```