Institute of Computer Technology

B. Tech. Computer Science and Engineering

Sub: ESFP – I

Assignment-1

Name:-Dwij desai Roll no.BDA08 class:-B Batch:- BDA

1. What are Tokens? Explain various kinds of tokens in the C programming language?

Assignment-I

Name:- Dwij Desai Roll No. BDA08

Sem-I Branch:- BDA

Q-1 What are token? Explain various kinds of token in the C programming language?

Ans In the context of programming language, "token" are the smallest units of a program that have meaning. In the C programming language, token are used to represent various elements of the code, such as keywords, indentifiers, constants, and operators. C uses whitespace to separate token, so spaces, tabs, and newlines are typically used to delimit tokens.

Here are the various kinds of tokens in the c programming language:-

① Keywords:-

Keywords are reserved words that have predefined meanings in c and cannot be used as indentifies (variable names, fuction names, etc.).

Dwij Desai BDA08

- Examples of C keywords include 'if', 'while', 'for', 'int', 'return', and 'struct'.

② Identifiers :-

Identifiers are used for naming variables, functions, and other user-defined entities character (a-z, A-z) or an underscore (_) and can be followed by letters, digits or underscores, for example, 'myvariable' and 'calculateArea' are indentifiers.

③ Constants :-

Constants are values that do not change during the program's execution.

- C supports serveral type of constants.
  - → Interger constants :-
    - These can be written in decimal, Octal, or hexadecimal notation.
  - → Floating-Point constants :-

Dwij Desai BDA08

- These include real numbers with a decimal point or in scientific notation.

→ Character constants :-

- These are single characters enclosed in single quotes, like 'A' or '7'.

→ String constants :-

- These are sequences of characters enclosed in double quotes, like "Hello".

④ Operators :-

Operators in C are used for performing various operations on variables and values.

- Common operators include ~~arithmetic~~

→ arithmetic operators (+, -, *, /, %)

→ relational operators (==, !=, <, >, <=, >=)

→ logical operators (&&, ||, !)

Dwij Desai BDA08

→ assignment operators (=, +=, -=, *=, /=, %=)

and more

⑤ Punctuation symbols :-

Punctuation symbols are characters that are used to structure and separate parts of the code.

- These include :
  - → semicolons ';'
  - → commas ','
  - → parentheses '()'
  - → square brackets '[]'
  - → curly braces '{}'
  - → period '.'
- For members access in structs and pointers.

⑥ Comments :-

Comments are not considered part of C program's executable code but provide explanatory notes to programmers. There are two types of comments in C :-

Here's an example of a simple C program with various tokens:

```
#include <stdio.h>

int main() {
    // This is a single-line comment
    int num1 = 10;        // Variable declaration and assignment
    int num2 = 20;

    if (num1 < num2) {     // Conditional statement
```

```
        printf("num1 is less than num2\n");
    } else {
        printf("num1 is greater than or equal to num2\n");
    }

    return 0;
}
```

Question – 2:

Explain the differences between compile time and run time errors with the help examples?

Dwij Desai BDA08

Q-2 Explain the diff. between compile time and run time errors with the help examples?

Ans

Compile time Error	Run-time Error
• They are also known as syntax error, occur when the complier found an incorrect statement.	• They are also known as execution or run time error, they occur when the complies could not detect any error.
• They errors are detected and reported by the during of compilation process.	• They are detected and handled during the execution.
• They are caused by syntax issues, like, missing semi-colons and man.	• They occur due to incorrect logic, incorrect data type.
• Eg:- printf("Hello, world!") printf("Hello, world)	• Eg:- def divid(a,b): return a/b print(divid(10,0)).

Example of a compile-time error:

```
int main() { int x = 5; printf("The value of y is %d\n", y); // Using an undeclared variable return 0; }
```

Example of a run-time error:

```
int main() { int a = 10; int b = 0; int result = a / b; // Division by zero printf("Result: %d\n", result); return 0; }
```

Dwij Desai BDA08

Q-3 C is a statically typed programming language, which means that you need to declare the data type of a variable before using it. C provides a variety of data types to work with diff. types of values. Here are some common data types in C:

① int: Used to store integer values.

② float: Used to store single-precision floating point numbers.

③ double: Used to store double-precision floating-point numbers.

④ Char: Used to store single character.

⑤ Bool: Used to store boolean values (0 for false, 1 for ture).

⑥ long: Used to store larger integers.

⑦ Short: Used to store short integers.

⑧ Unsigned: Used to store only positive integers (unsinged int, unsigned char).

Dwij Desai BDA08

Here's a simple C program that demonstrates the use of different data types:-

In this program, we declare and initialize variables of different data types and then use the 'printf' function to print their values.

You can see how each data type is used to store and display different kinds of values.

Here is an program for question 3:

```
#include <stdio.h>

int main() {
    int integerVar = 42;
    float floatVar = 3.1415;
    double doubleVar = 2.71828;
    char charVar = 'A';
    _Bool boolVar = 1; // true
    long longVar = 1234567890;
    short shortVar = 32767;
    unsigned int unsignedVar = 100;

    printf("int: %d\n", integerVar);
    printf("float: %f\n", floatVar);
    printf("double: %lf\n", doubleVar);
    printf("char: %c\n", charVar);
    printf("_Bool: %d\n", boolVar);
    printf("long: %ld\n", longVar);
    printf("short: %d\n", shortVar);
    printf("unsigned int: %u\n", unsignedVar);

    return 0;
}
```

Question – 4: Which variables below are syntactically correct?

Answer: This are the variables that are syntactically correct Num_var, var1 var2, v2 and I

The variables "10a," "float," "product's," and "r.no" are not syntactically correct:

10a - Variable names cannot start with a digit. "10a" starts with a digit, making it an invalid variable name.

float - "float" is not a valid variable name because it is a reserved keyword in C and cannot be used as an identifier.

product's - Variable names cannot contain special characters like apostrophes. "product's" contains an apostrophe, which is not allowed.

r.no - Variable names cannot contain periods. "r.no" contains a period, which is not allowed in variable names.

Question – 5: What will happen if a variable is declared as constant? Give a program example for constant variable declaration.

Q-5 If the any variable is declared as constant, then the value can't be updated or changed any where in the program.

Ans

```
#include <stdio.h>

using namespace std;
int main ()
{
    const double pi=3.14;
    cont << "Area of circle R4="<<pi*4*4;
    return 0;
}
```

Dwij Desai BDA08

Q-6 Here's an algorithm to input a character values prints it's ascii code and determine whether it's an alphabet or a digit:-

Ans ① Input:- Accept a character from the user.

② Convert to ASCII:-
Use the charater's ASCII value by typecasting it to an integer. Most programming languages have functions or operators for this purpose, like 'ord()' in python or ~~sy~~ simple typecasting in c.

③ Check for Alphabet or Digit:-

- If the ASCII value is within the range of lowercase alphabet characters or uppercase alphabet charaters. it's an alphabet.
- If the ASCII value is within the range of digit characters. it's a digit.

Dwij Desai BDA08

4) output:-

- Print the ASCII value.
- Print wether it's an alphabet or digit based on the checks in step-3.

Example of using arithmetic operators:

```
#include <stdio.h>

int main() {
    int a = 10;
    int b = 5;

    int sum = a + b;
    int difference = a - b;
    int product = a * b;
    int quotient = a / b;
    int remainder = a % b;

    printf("Sum: %d\n", sum);
    printf("Difference: %d\n", difference);
    printf("Product: %d\n", product);
    printf("Quotient: %d\n", quotient);
    printf("Remainder: %d\n", remainder);

    return 0;
}
```

Example of using relational operators:

```
    #include <stdio.h>

int main() {
    int x = 5;
    int y = 10;

    if (x == y) {
```

```
        printf("x is equal to y.\n");
    } else {
        printf("x is not equal to y.\n");
    }

    if (x < y) {
        printf("x is less than y.\n");
    }

    return 0;
}
```

Example of using logical operators:

```
#include <stdio.h>

int main() {
    int a = 1;
    int b = 0;

    if (a && b) {
        printf("Both a and b are true.\n");
    }

    if (a || b) {
        printf("At least one of a or b is true.\n");
    }

    if (!b) {
        printf("b is false (negation).\n");
    }

    return 0;
}
```

Example of using assignment operators:

```
#include <stdio.h>

int main() {
    int x = 10;
    x += 5;  // Equivalent to x = x + 5;
    printf("x is now: %d\n", x);

    return 0;
}
```

Question – 8: Explain the concept if, if – else, nested if – else and else if ladder condition in c with the help of program example.

Answer:

conditional statements allow you to make decisions in your code based on certain conditions. There are four main types of conditional statements: if, if-else, nested if-else, and else-if ladder. Each of these statements serves a different purpose in controlling the flow of your program. Here's an explanation of each with program examples:

if Statement:

The if statement is the most basic form of a conditional statement. It allows you to execute a block of code if a given condition is true.

if-else Statement:

The if-else statement allows you to execute one block of code if the condition is true and another block of code if the condition is false.

Nested if-else Statement:

You can have if-else statements inside other if or else blocks. This is known as nesting, and it allows for more complex decision-making.

else if Ladder:

An else if ladder is used when you have multiple conditions to check, and you want to execute different code blocks based on which condition is true.

Question – 9:

What are the classifications of loops? Explain with the help of syntax, flowchart, and suitable program examples?

Answer: There are three main types of loops that are used to execute a block of code repeatedly based on a condition

for loop:

The for loop is used when you know the number of iterations in advance. It consists of three parts: initialization, condition, and increment/decrement. The loop runs as long as the condition is true.

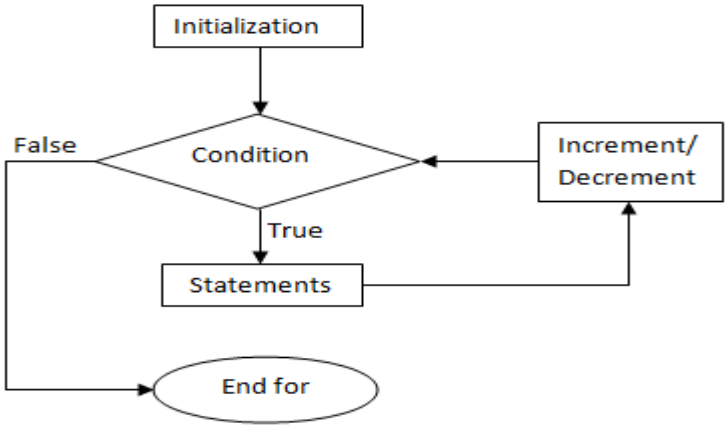
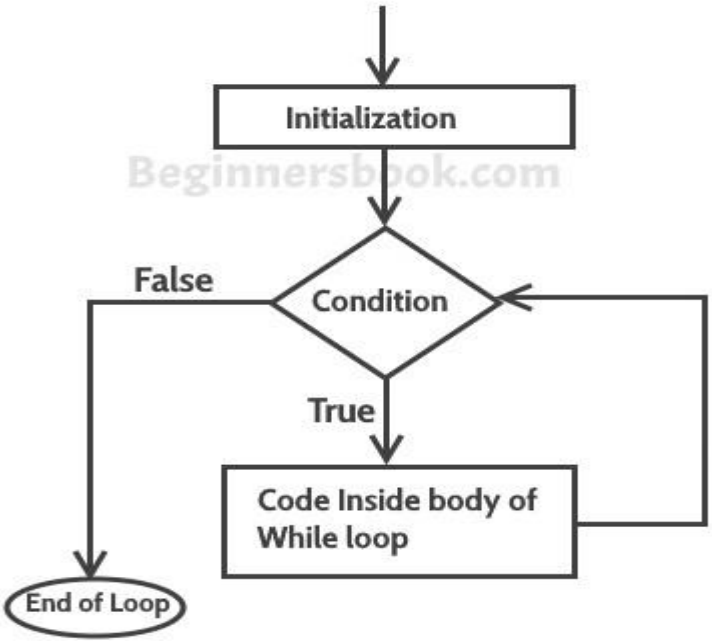


fig: Flowchart for for loop

while Loop:

The while loop is used when you don't know the number of iterations in advance, and it runs as long as a given condition is true.

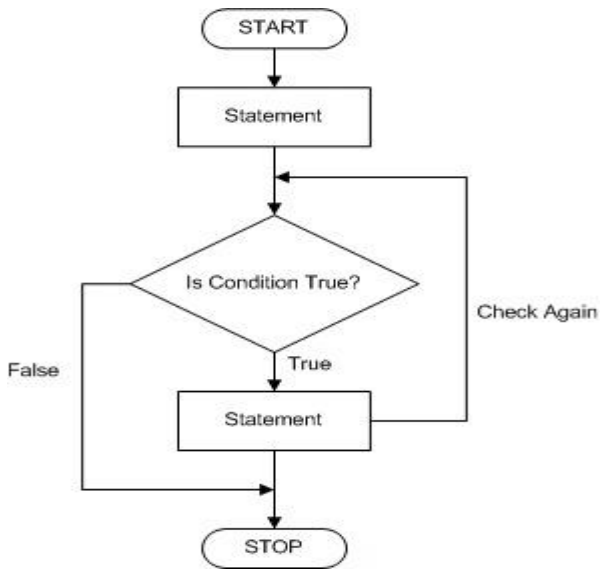
```
while (condition)
{
        // code to be executed
}
```



do-while Loop:

The do-while loop is similar to the while loop, but it guarantees that the code block is executed at least once. It checks the condition after executing the code block.

```
do
{
        // code to be executed
} while (condition);
```



Find Outputs of following programs:

1.

```
#include<stdio.h>
int main ()
{
    int p=5;
    printf("%d",++(p));
    return 0;
}
```

The code you provided contains a syntax error and won't compile successfully in C. The error is in the line ‘printf("%d",++(p)); ‘. The ‘(p) ‘ inside the ‘++ ‘ operator is incorrect; it should be ‘p ‘.

Output:6

2.

```
#include<stdio.h>

int main ()
{
    int i=10;

    i=!(i>14);
```

```
    printf ("i=%d",i);

    return 0;
}
```

‘i = !i > 14; ‘: This line contains two issues:

- The ‘! ‘ operator is used to perform logical negation, which results in a boolean value of ‘0 ‘ (false) or ‘1 ‘ (true). In this context, it will treat ‘i ‘ as a boolean expression.

- The comparison ‘!i > 14 ‘ is likely not what you intended. It will evaluate to either ‘0 ‘ or ‘1 ‘ (depending on the value of ‘i ‘), which is then compared to ‘14 ‘.

3.

```
#include<stdio.h>
int main()
{
    int num =10;
    printf("%10d", num);

    return 0;
}
```

‘printf("%10d", num); ‘: This line uses the ‘printf ‘ function to format and print the value of ‘num‘

. Here's the explanation:

- ‘printf ‘ is a function used for formatted output in C.
- ‘"%10d" ‘ is a format specifier:
- ‘% ‘ indicates that a format specifier follows.
- ‘10 ‘ specifies the field width. It means that the printed value should be right-aligned within a field of 10 characters.
- ‘d ‘ specifies that the argument to be printed is an integer (‘num ‘ in this case).

4.

```
#include<stdio.h>
#define a 10
int main ()
{
    printf("%d",a);

    return 0;
```

```
}
```

Output:10

```
[Running] cd "e:\ESFP\" && gcc assignment1.c -o assignment1 && "e:\ESFP\"assignment1
10
[Done] exited with code=0 in 0.247 seconds
```

5.

```c
#include<stdio.h>
int main ()
{
    int i=10,j=20;
    printf("%d..%d",i,j);

    return 0;
}
```

Output:10..20

```
[Running] cd "e:\ESFP\" && gcc assignment1.c -o assignment1 && "e:\ESFP\"assignment1
10..20
[Done] exited with code=6 in 0.248 seconds
```

6.

```c
#include <stdio.h>
int main()
{
    int i;
    printf("%d", scanf("%d", &i));
    return 0;
}
```

7.

```c
#include <stdio.h>
void main()
{
    char not;
    not = !2;
    printf("%d", not);
}
```

Programing exercise :

1.

```c
#include <stdio.h>

int main()
{
    int num1, num2, num3, least;

    printf("Enter three integers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    least = (num1 < num2) ? ((num1 < num3) ? num1 : num3) : ((num2 <
num3) ? num2 : num3);

    printf("The least number is: %d\n", least);

    return 0;
}
```

```
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals> cd "c:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1\" ; if ($?) { gcc x.c -o x } ; if
 ($?) { .\x }
Enter three integers: 123
1
1
The least number is: 1
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1>
```

2.

```c
#include <stdio.h>

int isPrime(int num)
{
    if (num <= 1)
    {
        return 0;
```

```
    }
    for (int i = 2; i * i <= num; i++)
    {
        if (num % i == 0)
        {
            return 0;
        }
    }
    return 1;
}

int isPerfect(int num)
{
    int sum = 0;
    for (int i = 1; i <= num / 2; i++)
    {
        if (num % i == 0)
        {
            sum += i;
        }
    }
    return (sum == num);
}

int main()
{
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num % 2 == 0)
    {
        printf("%d is even.\n", num);
    }
    else
    {
        printf("%d is odd.\n", num);
    }
```



```
    if (isPrime(num))
    {
        printf("%d is a prime number.\n", num);
    }
    else
```

```
    {
        printf("%d is not a prime number.\n", num);
    }

    if (isPerfect(num))
    {
        printf("%d is a perfect number.\n", num);
    }
    else
    {
        printf("%d is not a perfect number.\n", num);
    }

    return 0;
}
```

3.

```
#include <stdio.h>

int main()
{
    float tCharge, uCharge, tax;
    int unit;


    printf("Enter monthly electricity consumed units: ");
    scanf("%d", &unit);

    if (unit >= 1 && unit <= 100)
    {
        uCharge = 5.0;
    }
    else if (unit >= 101 && unit <= 200)
    {
        uCharge = 10.0;
    }
    else if (unit >= 201 && unit <= 300)
    {
        uCharge = 15.0;
    }
    else
```

```c
    {
        uCharge = 20.0;
    }

    tCharge = unit * uCharge;
    tax = 0.05 * tCharge;
    tCharge += tax;
    printf("Total monthly electricity charge: Rs. %.2f\n", tCharge);
    return 0;
}
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS                                Code

PS C:\Users\dwijd\Downloads\esfp-1 Prcticals> cd "c:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1\" ; if ($?) { gcc x.c -o x } ; if
 ($?) { .\x }
Enter monthly electricity consumed units: 234
Total monthly electricity charge: Rs. 3685.50
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1>
```

4.

```c
#include <stdio.h>

int main()
{
    float bSalary,da, hra, others, pf, grossSalary, ma, netSalary;

    printf("Enter basic salary: Rs. ");
    scanf("%f", &bSalary);

    da=0.12*bSalary;
    hra=0.30*bSalary;
    others=1000.0;
    pf=0.12*bSalary;
    grossSalary=(bSalary + da + hra)-pf;
    ma = 0.05*bSalary;
    netSalary=grossSalary - ma;
    printf("Net Salary: Rs. %.2f\n", netSalary);
    return 0;
}
```

```
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals> cd "c:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1\" ; if ($?) { gcc x.c -o x } ; if
 ($?) { .\x }
Enter basic salary: Rs. 3245
Net Salary: Rs. 4056.25
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1>
```

5.

```c
#include <stdio.h>

int main()
{
    int num;
    int sum=0;
    int product=1;

    printf("Enter a five-digit number: ");
    scanf("%d", &num);

    if (num < 10000 || num > 99999)
    {
        printf("Invalid input. Please enter a five-digit number.\n");
        return 0;
    }

    int digit;
    int alternate = 1;

    while (num > 0)
    {
        digit = num % 10;
        if (alternate)
        {
            product *= digit;
        }
        else
        {
            sum += digit;
        }
        alternate = 1 - alternate;
        num /= 10;
    }

    printf("Multiplication = %d\n", product);
    printf("Addition = %d\n", sum);

    return 0;
}
```

```
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals> cd "c:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1\" ; if ($?) { gcc x.c -o x } ; if
 ($?) { .\x }
Enter a five-digit number: 12345
Multiplication = 15
Addition = 6
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1>
```

6.

```c
#include <stdio.h>

int main()
{
    int i, j, k;

    for (i = 6; i >0; i--)
    {
        char ch = 'A';
        for (j = 0; j < i; j++)
        {
            printf("%c", ch);
            ch++;
        }
        for (k = 0; k < 2*(6 - i); k++)
        {
            printf(" ");
        }
        ch--;
        for (j = 0; j < i; j++)
        {
            printf("%c", ch);
            ch--;
        }
        printf("\n");
    }

    printf("\n");

    for (i = 0; i < 5; i++)
    {
        for(j=0;j<i;j++)
        {
            printf(" ");
        }
        for (j = 0; j < 9 - 2 * i; j++)
        {
            printf("*");
```

```
        }
        printf("\n");
    }

    return 0;
}
```

```
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals> cd "c:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1\" ; if ($?) { gcc x.c -o x } ; if
 ($?) { .\x }
ABCDEFFEDCBA
ABCDE   EDCBA
ABCD     DCBA
ABC       CBA
AB         BA
A           A

*********
 *******
  *****
   ***
    *
PS C:\Users\dwijd\Downloads\esfp-1 Prcticals\ESFP-1_Practicals_codes\Assignment-1>
```