EXPERIMENT NO. 03

DATE OF PERFORMANCE: **GRADE:**

DATE OF ASSESSMENT: **SIGNATURE OF LECTURER/ TTA:**

AIM: To Study Token, Expression and Control Structures.

THEORY:

Tokens: **The smallest individual unit in a program is known as token. C++ has the following tokens.**

- **Keywords**
- **Identifiers**
- **Constants**
- **Strings**
- **Operators**

Keywords: **The keywords implement specific C++ language features. They are explicitly reserved identifiers which cannot be used as names for the program variables or other user-defined program elements.**

Many of which are common to the keywords of C++ like for, friend, goto, if, switch etc..

Constants:

Constants refer to fixed values that do not change during the course of execution of the program.

Variable:

- **C++ variable is a named location in a memory where a program can manipulate the data. This location is used to hold the value of the variable.**
- **The value of the C++ variable may get change in the program.**
- **C++ variable might be belonging to any of the data type like int, float, char etc.**

Data Types:

Data types in C++ can be broadly categorised into three types :-

- **User Defined**
- **Built In**
- **Derived Data type**

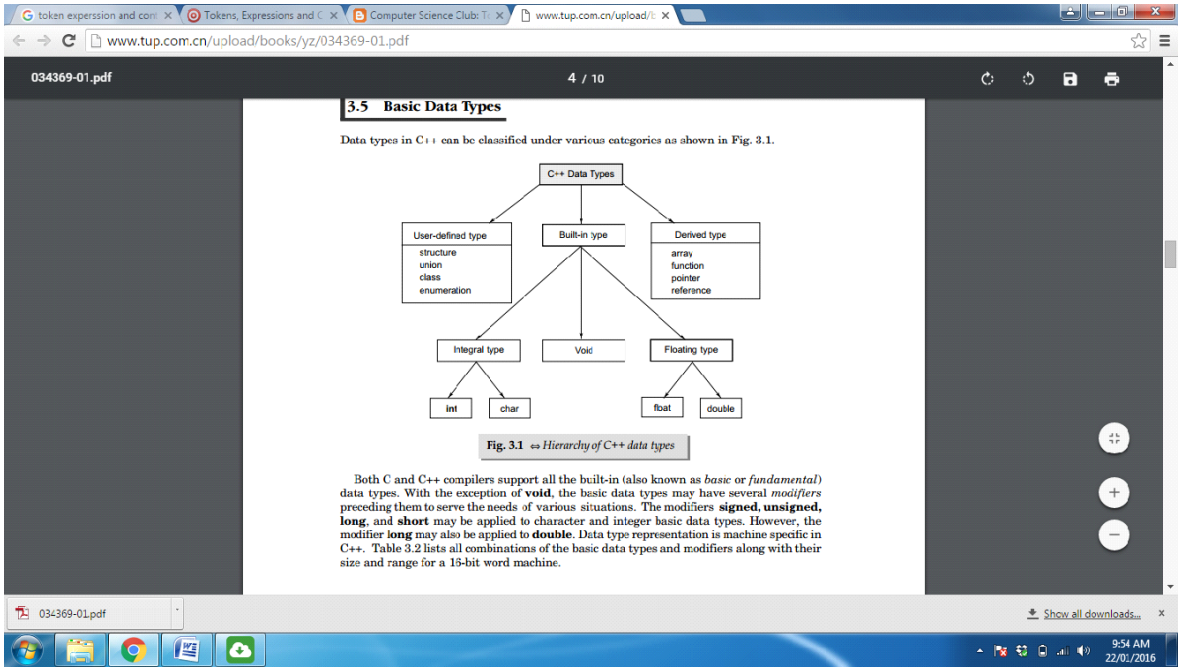
Identifiers and Constants:

- **Each program elements in a C++ program are given a name called identifiers.**
- **Names given to identify Variables, functions and arrays are examples for identifiers. eg. x is a name given to integer variable in program.**

Rules for constructing identifier name in C++:

- **Only alphabetic characters, digits and underscores are permitted.**
- **The name cannot start with a digit.**
- **Uppercase and lowercase letters are distinct.**
- **A declared keyword cannot be used as a variable name.**

Basic Data Types:



C++ Operators and Expressions:

- **The symbols which are used to perform logical and mathematical operations in a C ++ program are called operators.**
- **These C++ operators join individual constants and variables to form expressions.**
- **Operators, constants and variables are combined together to form expressions.**

- **Consider the expression A + B * 5 where, +, * are operators, A, B are variables, 5 is constant and A + B * 5 is an expression.**

Types of C++ operators:

C++ language offers many types of operators. They are,

- **Arithmetic operators**
- **Assignment operators**
- **Relational operators**
- **Logical operators**
- **Bit wise operators**
- **Conditional operators (ternary operators)**
- **Increment/decrement operators**
- **Special operators**

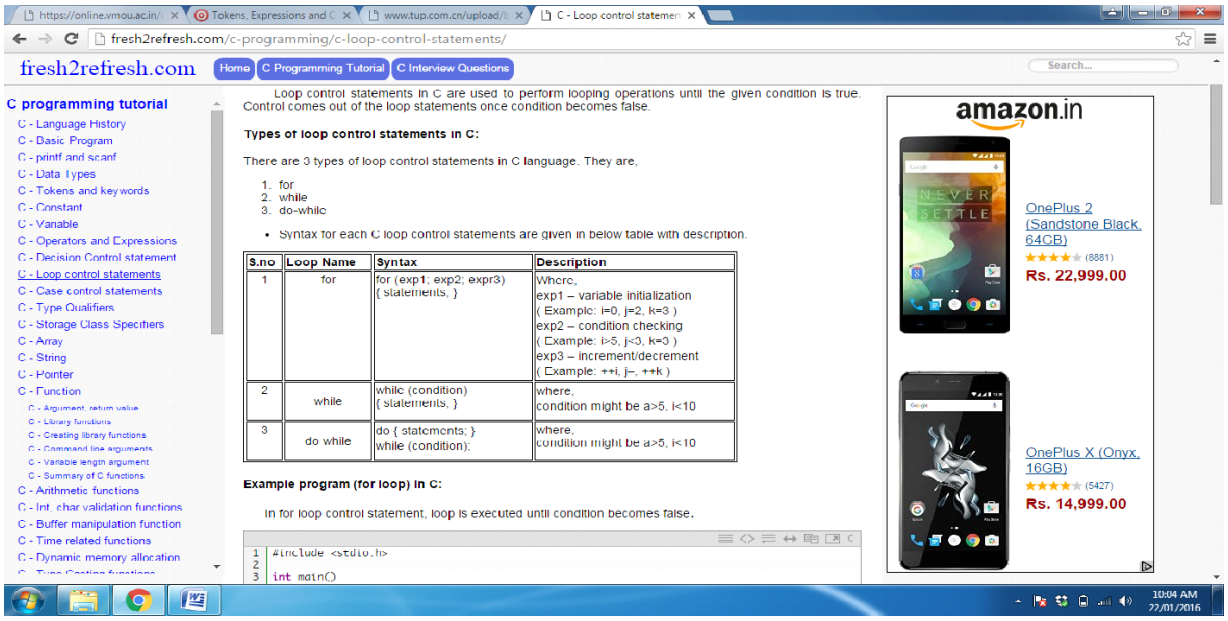
C++ Decision Control statement and Loop:

- **In decision control statements (C++ if else and nested if), group of statements are executed when condition is true. If condition is false, then else part statements are executed.**
- **There are 4 types of decision making control statements in C++ language. They are,**
  - **if statements**
  - **if else statements**
  - **nested if statements**
  - **switch case statements**

Loop:

There are 3 types of loop control statements in C++ language. They are,

- **for**
- **while**
- **do-while**



<u>Enumeration</u>: - enumerated types is to create new data types that can take on only a restricted range of values.

Moreover, these values are all expressed as constants rather than magic numbers--in fact, there should be no need to know the underlying values. The names of the constants should be sufficient for the purposes of comparing values.

Example:-

enum wind_directions_t {NO_WIND, NORTH_WIND, SOUTH_WIND, EAST_WIND, WEST_WIND};

Reference variable:

A reference variable is an alias, that is, another name for an already existing variable. Once a reference is initialized with a variable, either the variable name or the reference name may be used to refer to the variable.

Example:

Manipulator:

Manipulators are helper functions that make it possible to control input/output streams using operator<< or operator>>. The manipulatorsthat are invoked without arguments. They are implemented as functions that take a reference to a stream as their only argument.

Manipulator functions are special stream functions that change certain characteristics of the input and output. They change the format flags and values for a stream. The main advantage of using manipulator functions is that they facilitate that formatting of input and output streams.

To carry out the operations of these manipulator functions in a user program, the header file input and output manipulator <iomanip.h> must be included.

(a) Endl:- **the endl is an output manipulator to generate a carriage return or line feed character. The endl may be used several times in a C++ statement.**

For example,

(1)

```
cout << " a " << endl << "b" << endl;
```

(2)

```
cout << " a = " << a << endl;
cout << " b = " << b << endl;
```

A program to display a message on two lines using the endl manipulator and the corresponding output is given below.

```
/ / using endl manipulator
#include <iostream.h>
Void main (void)
```

```
{
    cout << " My name is computer ";
    cout << endl;
    cout << " many greetings to you ";
}
```

Output of the above program

My name is computer
Many greetings to you

(b) Setw () :- The setw ( ) stands for the set width. The setw ( ) manipulator is used to specify the minimum number of character positions on the output field a variable will consume.

The general format of the setw manipulator function is
setw(int w)

Which changes the field width to w, but only for the next insertion. The default field width is 0.
For example,

```
cout << setw (1) << a << endl;
cout << setw (10) << a << endl;
```

A program to display the data variables using setw manipulator functions.

```
/ /using setw manipulator
#include <iostream.h>
#include <iomanip.h>
void main (void)
{
   int  a,b;
   a = 200;
   b = 300;
  cout << setw (5) << a << setw (5) << b << endl;
  cout << setw (6) << a << setw (6) << b << endl;
  cout << setw (7) << a << setw (7) << b << endl;
  cout << setw (8) << a << setw (8) << b << endl;
}
```

Output of the above program

```
   200         300
     200         300
       200         300
         200         300
```

PROGRAM 1: Use of manipulators in C++

```
#include <iostream>

#include <iomanip>


int main()

{

   int basic = 950, allowance = 95, total = 1095;


   std::cout << std::setw(10) << "Basic" << std::setw(10) << basic << std::endl;

   std::cout << std::setw(10) << "Allowance" << std::setw(10) << allowance << std::endl;

   std::cout << std::setw(10) << "Total" << std::setw(10) << total << std::endl;


   return 0;

}
```

OUTPUT:

```
    Basic       950

  Allowance        95

    Total      1095
```

PROGRAM 2: Reference Variable in C++

```
#include <iostream>
using namespace std;

int main()
```

```
{
   // declare simple variables
   int i;
   double d;

   // declare reference variables
   int& r = i;
   double& s = d;

   i = 5;
   cout << "Value of i: " << i << endl;
   cout << "Value of i reference: " << r << endl;

   d = 11.7;
   cout << "Value of d: " << d << endl;
   cout << "Value of d reference: " << s << endl;

   return 0;
}
```

Output:

Value of i: 5
Value of i reference: 5
Value of d: 11.7
Value of d reference: 11.7

PROGRAM 3: Reference Variable as function argument in C++

```
#include <iostream>


void f(int& x)

{

   x = x + 10;

   std::cout << "Value of M is Now " << x;

}
```

```
int main()
{
   int m = 10;

   std::cout << "Value of M is " << m << "\n";
   f(m);

   return 0;
}
```

OUTPUT:

Value of M is 10

Value of M is Now 20

Program 4: Scope Resolution operator in C++

```
#include <iostream>

int m = 10;

int main()
{
   int m = 20;
   {
      int k = m;
```

```
        int m = 30;
        std::cout << "We are in Inner Block\n";
        std::cout << "K = " << k << "\n";
        std::cout << "M = " << m << "\n";
        std::cout << "::M = " << ::m << "\n";
    }
    std::cout << "\nWe are in Outer Block\n";
    std::cout << "M = " << m << "\n";
    std::cout << "::M = " << ::m << "\n";

    return 0;
}
```

OUTPUT:

We are in Inner Block

K = 20

M = 30

::M = 10

We are in Outer Block

M = 20

::M = 10

PROGRAM 5: Create an Enum of a Shape having Circle, rectangle and triangle in C++.

```
#include <iostream>

enum Shape
{
   circle,
   rectangle,
   triangle
};

int main()
{
   int code;

   std::cout << "Enter Shape Code: ";
   std::cin >> code;

   while (code >= circle && code <= triangle)
   {
      switch (code)
      {
         case circle:
            std::cout << "Circle\n";
            break;
         case rectangle:
            std::cout << "Rectangle\n";
```

```
            break;
        case triangle:
            std::cout << "Triangle\n";
            break;
    }

    std::cout << "Enter Shape Code: ";
    std::cin >> code;
  }

  std::cout << "Bye\n";

  return 0;
}
```

OUTPUT:

```
Enter Shape Code: 1
Circle
Enter Shape Code: 2
Rectangle
Enter Shape Code: 3
Triangle
Enter Shape Code: 4
Bye
```

PROGRAM 6: program to show use of if else statement

```
#include <iostream>
```

```
using namespace std;


int main()
{
   int number;
   cout << "Enter an integer: ";
   cin >> number;


   // checks if the number is positive
   if (number > 0)
   {
      cout << "You entered a positive integer: " << number << endl;
   }


   cout << "This statement is always executed.";
   return 0;
}
```

OUTPUT:

```
Enter an integer: 5
You entered a positive integer: 5
This statement is always executed.
```

PROGRAM 7: program to show use of else if ladder

```
#include <iostream>
```

```
using namespace std;

int main()
{
   // declare local variable
   int marks = 55;

   // check the boolean condition
   if (marks >= 80)
   {
      // if 1st condition is true
      cout << "U are 1st class !!" << endl;
   }
   else if (marks >= 60 && marks < 80)
   {
      // if 2nd condition is true
      cout << "U are 2nd class !!" << endl;
   }
   else if (marks >= 40 && marks < 60)
   {
      // if 3rd condition is true
      cout << "U are 3rd class !!" << endl;
   }
   else
   {
```

```
        // none of the conditions are true
        cout << "U are fail !!" << endl;
    }

    return 0;
}
```

OUTPUT:

U are 3rd class !!

PROGRAM 8: Program to build a simple calculator using switch Statement.

```
#include <iostream>
using namespace std;

int main()
{
    char o;
    float num1, num2;

    cout << "Enter an operator (+, -, *, /): ";
    cin >> o;

    cout << "Enter two operands: ";
    cin >> num1 >> num2;

    switch (o)
    {
        case '+':
            cout << num1 << " + " << num2 << " = " << num1 + num2;
            break;
        case '-':
            cout << num1 << " - " << num2 << " = " << num1 - num2;
```

```
        break;
    case '*':
        cout << num1 << " * " << num2 << " = " << num1 * num2;
        break;
    case '/':
        cout << num1 << " / " << num2 << " = " << num1 / num2;
        break;
    default:
        // operator doesn't match any case constant (+, -, *, /)
        cout << "Error! Operator is not correct";
        break;
  }

  return 0;
}
```

OUTPUT:

Enter an operator (+, -, *, /): +

Enter two operands: 5 3

5 + 3 = 8

PROGRAM 9: Program to find factorial of a number by using for loop.

```
#include <iostream>
using namespace std;

int main()
{
  int i, n, factorial = 1;

  cout << "Enter a positive integer: ";
```

```
    cin >> n;

    for (i = 1; i <= n; ++i) {
        factorial *= i;   // factorial = factorial * i;
    }

    cout << "Factorial of " << n << " = " << factorial;
    return 0;
}
```

OUTPUT:

```
Enter a positive integer: 5

Factorial of 5 = 120
```

PROGRAM 10: Program to compute factorial of a number by using while loop.

```
#include <iostream>

using namespace std;


int main()

{

    int number, i = 1, factorial = 1;


    cout << "Enter a positive integer: ";

    cin >> number;


    while (i <= number) {

        factorial *= i;      // factorial = factorial * i;
```

```
    ++i;
  }

  cout << "Factorial of " << number << " = " << factorial;
  return 0;
}
```

OUTPUT:

```
Enter a positive integer: 5
Factorial of 5 = 120
```

PROGRAM 11: Program to add numbers until user enters 0 by using do while loop.

```
#include <iostream>
using namespace std;

int main()
{
  float number, sum = 0.0;

  do {
    cout << "Enter a number: ";
    cin >> number;
    sum += number;
  }
  while (number != 0.0);

  cout << "Total sum = " << sum;
```

```
    return 0;
}
```

OUTPUT:

```
Enter a number: 2.5
Enter a number: 3.1
Enter a number: 0
Total sum = 5.6
```

PROGRAM 12: program to calculate the average of numbers entered by user by using goto statement.

```
#include <iostream>
using namespace std;

int main()
{
    float num, average, sum = 0.0;
    int i, n;

    cout << "Maximum number of inputs: ";
    cin >> n;

    for (i = 1; i <= n; ++i)
    {
        cout << "Enter n" << i << ": ";
        cin >> num;

        if (num < 0.0)
        {
            // Control of the program moves to jump:
            goto jump;
        }
        sum += num;
    }
```

```
jump:
   average = sum / (i - 1);
   cout << "\nAverage = " << average;
   return 0;
}
```

OUTPUT:

Maximum number of inputs: 5

Enter n1: 2

Enter n2: 3

Enter n3: -1

Average = 2.5

PROGRAM 13: Program to demonstrate working of break statement.

```
#include <iostream>
using namespace std;

int main() {
   float number, sum = 0.0;

   // test expression is always true
   while (true)
   {
      cout << "Enter a number: ";
      cin >> number;

      if (number != 0.0)
      {
         sum += number;
      }
      else
```

```
        {
            // terminates the loop if number equals 0.0
            break;
        }

    }
    cout << "Sum = " << sum;

    return 0;
}
```

OUTPUT:

```
Enter a number: 5
Enter a number: 3.2
Enter a number: 1
Enter a number: 0

Sum = 9.2
```