SRIMAAN COACHING CENTRE-TRICHY-PG-TRB- COMPUTER INSTRUCTOR GRADE-1 PROGRAMMING IN C++ UNIT-I-STUDY MATERIAL-TO CONTACT:8072230063.

2023-24 SRIMAAN

SRIMAAN

PG-TRB

COMPUTER

INSTRUCTOR GRADE-I

UNIT-5- PROGRAMMING IN C++

(NEW SYLLABUS 2023-24)

PG-TRB STUDY MATERIALS:–TAMIL/ENGLISH/ MATHEMATICS/PHYSICS CHEMISTRY/COMMERCE (T/M & E/M)/BOTANY (T/M & E/M)/ ZOOLOGY HISTORY (T/E)/ECONOMICS (T/E)/ GEOGRAPHY /BIO-CHEMISTRY PGTRB-COMPUTER INSTRUCTOR GRADE-I -TO CONTACT -8072230063.

SRIMAAN COACHING CENTRE-TRICHY-PG-TRB- COMPUTER INSTRUCTOR GRADE-1 PROGRAMMING IN C++ UNIT-I-STUDY MATERIAL-TO CONTACT:8072230063.

2023-24 SRIMAAN

PG-TRB MATERIALS

TAMIL/MATHS / ENGLISH/ COMMERCE/ CHEMISTRY/ PHYSICS/BOTANY/ ZOOLOGY/ HISTORY/ECONOMICS/ AVAILABLE.

TRB POLYTECHNIC LECTURER MATERIALS:

MATHEMATICS / ENGLISH /PHYSICS / CHEMISTRY/ COMPUTER SCIENCE / IT / EEE /ECE /EIE/ICE MECHANICAL/CIVIL/MOP AVAILABLE.

TNEB-ASSESSOR/AE/JA MATERIALS WITH QUESTION BANK AVAILABLE.

BEO & DEO (TAMI & ENGLISH MEDIUM) FULL STUDY MATERIAL WITH QUESTION BANK AVAILABLE.

TET /PG-TRB /TRB-POLYTECHNIC/ BEO & DEO MATERIALS ARE SENDING THROUGH COURIER.

TO CONTACT

8072230063

PG-TRB STUDY MATERIALS:-TAMIL/ENGLISH/ MATHEMATICS/PHYSICS CHEMISTRY/COMMERCE (T/M & E/M)/BOTANY (T/M & E/M)/ ZOOLOGY HISTORY (T/E)/ECONOMICS (T/E)/ GEOGRAPHY /BIO-CHEMISTRY PGTRB-COMPUTER INSTRUCTOR GRADE-I -TO CONTACT -8072230063.

SRIMAAN COACHING CENTRE-TRICHY.

TO CONTACT: 8072230063.

PG-TRB: COMPUTER INSTRUCTOR GRADE-I

SRIMAAN

UNIT-5

PROGRAMMING IN C++

SRIMAAN

NEW SYLLABUS-2023-24

The origin of C++:

C++ is an expanded version of **C. Bjarne Stroustrup** first invented the C++ extensions to C in 1980 at **Bell Laboratories** in **Murray Hill**, **New Jersey**. He initially called the new language as "C with Classes". However, in **1983 the name was changed to C++.**

Although C++'s predecessor, C, is one of the most liked and widely used professional languages in the world, the invention of C++ was necessitated by one major programming factor: increasing complexity. Over the years, computer programs have become larger and more complex.

Even though C is an excellent programming language, it too has its own limits. In C, once a program exceeds from25,000 to 100,000 lines of code, it becomes so complex that it is difficult to grasp as a totality.

- The purpose of C++ is to allow this barrier to be broken.
- The essence of C++ is to allow the programmer to comprehend and manage larger, more complex programs.
- Most additions made by Stroustrup to C support object-oriented programming sometimes referred to as OOP.

Definition of OOP:

Object oriented programming is a programming methodology that associates data structures with a set of operators, which act upon it.

OOP is a method of implementation in which programs are organized as cooperative collections of

objects, each of which represents an instance of some class and whose classes are all members of a hierarchy of classes united through the property called **inheritance.**

Depending on the object features supported, the languages are classified into two categories:

- Object-Based Programming Languages
- Object-Oriented Programming Languages
- Object-based programming languages support encapsulation object identity without
- supporting the important features of inheritance, polymorphism and message communications. Example :ADA.

Object-based language = Encapsulation + Object Identity

Object-Oriented Programming Language incorporate all the features of object-based programming languages along with **inheritance and polymorphism.**

Object-oriented programming language = Object
= based language + polymorphism +inheritance.

Introduction to Programming Languages:

The shift in programming language is categorized as following:

- Monolithic Programming
- Procedural Programming
- Structural Programming
- Object Oriented Programming

Monolithic Programming (Assembly language and BASIC):

This programming consists only global data and sequential code. Program flow control is achieved through the use of jump and the program code is duplicated each time it is used.

No subroutine concept is used. Since this programming style is not supporting the concept of data abstraction it is very difficult to maintain or enhance the program code.

Procedural Programming (FORTRAN and COBOL):

- Mainly comprises of algorithms.
- Programs were considered as important intermediate points between the problem and the computer in mid 1960s.
- Subprograms were originally seen as lab or saving devices but very quickly appreciated as a way to abstract program functions.

The important features of Procedural Programming are

- ✓ Programs are organized in the form of subroutines and all data items are global.
- ✓ Program controls are through jumps (goto's) and call subroutines.
- ✓ Abstracted subroutines are used to avoid repetition.
- ✓ Software application is minimized.
- ✓ Difficult to maintain and enhance the program code.

Structured programming (Pascal and C)

Structured programming is evolved as a mechanism to address the growing issues of programming in the large. Larger programming projects consist of large development teams, developing different parts of the same project independently.

Programs consist of multiple and in turn each module has a set of functions of related types.

- ♣ Structured programming is based upon the algorithm rather than data.
- ♣ Programs are divided into individual modules that perform different task.
- ♣ Controls the scope of data.
- ♣ Support modular programming.
- ♣ Introduction of user defined data types.

Advantages of C++

❖ Data abstraction :

In OOP, the data abstraction is defined as a collection of data and methods (functions).

❖ Data hiding :

In C++, the class construct allows to declare data and member functions, as a public, private and protected group. The implementation details of a class can be hidden. This is done by the data hiding principle.

- ❖ Hides internal object details (data members).
- ❖ Data hiding ensures exclusive data access to class members and protects object integrity by preventing unintended or intended changes.

❖ Data encapsulation :

The internal data (the member data) of a class are first separated from the outside world(the defined class). They are then put along with the member functions in a capsule. In other words, encapsulation groups all the pieces of an object into one neat package.

It avoids undesired side effects of the member data when it is defined out of the class and also protects the intentional misuse of important data. Classes efficiently manage the complexity of large programs through encapsulation.

- Means of data hiding
- Binds together code and data it manipulates and keeps both safe from outside interference.
- Tells exactly what user can access and can not access through public and private access specifiers .
- Prevents hacking of code.
- Combine data and operations on that data into a single unit (E.g. a class w/ public and private aspects).

Data abstraction and encapsulation:

- The wrapping up of data and methods into a single unit (called class) is known as **encapsulation.**
- Data encapsulation is the most striking feature of a class.
  The data is not accessible to the outside world and only those methods, which are wrapped in the class, can access it.
- These methods provide the interface between the object's data and the program. This insulation of the data from direct access by the program is called **data hiding.**
- Encapsulation makes it possible for objects to be treated like"black boxes" each performing a specific task without any concern for internal implementation.

Information in-Data & method- Information out

Abstraction refers to the act of representing essential features without including the background details or explanations. Classes use the concept of abstraction and are defined as a list of abstract attributes such as size, weight and cost and methods that operated on these attributes.

Classes:

Classes are created using the keyword class. A class declaration defines a new type that links code and data. This new type is then used to declare objects of that class.

Thus, a class is a logical abstraction, but an object has physical existence. In other words, an object is an instance of a class.

- Class represents a group of similar objects.

- A class is a way to bind the data describing an entity and its associated function together.
- Consider an account having characteristics n Account no., type, balance.
- Its associated operations are DEPOSITE and WITHDRAWAL.
- Each object is associated with the data of type class with which they are created.

Class:

The building block of C++ that leads to Object Oriented programming is a **Class.**
It is a user defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

For Example:

Consider the Class of **Cars**. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 **wheels, Speed Limit, Mileage range** etc.
So here, Car is the class and wheels, speed limits, mileage are their properties.

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to
- Manipulate these variables and together these data members and member functions defines the properties and behavior of the objects in a Class.
- In the above example of class *Car*, the data member will be **speed limit, mileage** etc and member functions can be **apply brakes**, **increase speed** etc.

An **Object** is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

Declaring Objects:

When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create objects.

Syntax:

Class Name Object Name;

EXAMPLE:

Given the declaration of class Human, the code segment
Human latvians[3600000];
defines an array latvians that has 3,600,000 elements, each of type Human.

Classes and objects :

- Class- user defined data type. Fundamental packaging unit of OOP technology.
- Class declaration is similar to struct declaration.
- Keyword 'class' followed by class name.
- Object is an instance of class.
- Object combines data and functions.
- Object is created as a variable of class type using class name.
- Members of class
  - (I) Data members / attributes
  - (II) Member functions / methods

Abstract Class

- Contains at least one pure virtual function.
- Object of abstract class can not be created, because it contains one or more pure virtual functions without definition.
- A reference or pointer can be created to support run time polymorphism.
- All the pure virtual functions of abstract class must be overridden in derived class.
- Can be used to create generic, extensible libraries for programmer to use in their own implementations.

Characteristics of Abstract Class

- Abstract class cannot be instantiated, but pointers and refrences of Abstract class type can be created.
- Abstract class can have normal functions and variables along with a pure virtual function.
- Abstract classes are mainly used for Upcasting, so that its derived classes can use its interface.
- Classes inheriting an Abstract Class must implement all pure virtual functions, or else they will become Abstract too.

Polymorphism

In OOP, polymorphism is defined as how to carry out different processing steps by a function having the same messages. Polymorphism treats objects of related classes in a generic manner.

- Polymorphism plays an important role in allowing objects having internal structures to share the same external interface.
- This means that a general class of operations may beϖ accessed in the same manner even though specific actions associated with each operations may differ.
- Polymorphism is extensively used in implementingϖ inheritance.
- **Polymorphism** is another important OOP concept. Polymorphism means the ability to take more than one form. For example, an operation may exhibit different behavior in different instances.
- The behavior depends upon the types of data used in the operation.
- For example, consider the operation of addition. For two numbers, the operation will generate a sum. If the operands were strings, then the operation would produce a third string by concatenation.
- Polymorphism plays an important role in allowing objects having different internal structures to share the same external interface.
- This means that a general class of operations may be accessed in the same manner even though

specific actions associated with each operation may differ. Polymorphism is extensively used in implementing inheritance.

Example:

```
//sample program for polymorphism

# include <iostream.h>
int abs(int I);
long abs (long l);
double abs(double d);
main()
{
        cout<<abs(-10)<<"\n";
        cout<<abs(10L)<<"\n";
        cout<<abs(-12.35)<<"\n";
return 0;
}
int abs(int i)
{
        cout<<"using int method";
        return i>0?i:-i;
}
long abs(long l)
{
        cout<<"using long method";
        return l>0?l:-l;
}
double abs(double d)
{
        cout<<"using double method";
        return d>0?d:-d;
}
```

Function Overloading:

Overloading refers to the use of the same thing for different purposes. C++ also permits overloading of functions. This means that we can use the same function name to create functions that perform a variety of different argument tasks. This is known as **function polymorphism** in OOP.

- Using the concept of function overloading, we can design a family of functions with one function name but with different argument lists.
- The function would perform different operations depending on the argument list in the function call.

- The correct function to be invoked is determined by checking the number and type of the arguments but on the function type.

For example, an overloaded add( ) function handles different types of data as shown below :

```
// Declarations
int add(int a, int b);
int add(int a,int b, int c);
double add(double x,double y);
double add(int p, double q);
double add(double p,int q);

// Function calls
cout << add(5,10);
cout << add(15,10.0);
cout << add(12.5,7.5);
```

Inheritance :

C++ allows a programmer to build hierarchy of classes. The derivation of classes is used for building hierarchy. The basic features of classes (parent or base classes) can be passed onto the derived classes.(child classes).

In practice, the inheritance principle reduces the amount of writing; as the derived classes do not have to be written again.

- A way of defining interfaces, re-using classes and extending original functionality.

- Inheritance is a process by which objects of one class acquire the properties of objects of another class.
- Allows a new class to inherit all the data members and member functions from a previously defined class.
- Works from more general objects to more specific objects.
- This is possible by deriving a new class from existing one. The new class will have the combined features of both the classes.
- Inheritance is the process of forming a new class from an **existing class** (or) **base class.**

Advantages of OOP:

- Through inheritance we can eliminate redundant code and extend the use of existing classes.
- We can build programs from the standard working modules that communicate with one another rather than having to start writing the code from scratch. This leads to saving of development time and higher

productivity.

- The principle of data hiding helps the programmer to build secure programs that cannot be invaded by code in other parts of the program.
- It is possible to have multiple objects to coexist without any interference.
- It is easy to partition the work in a project based on objects.
- The data centered design approach enables us to capture more details of a model in an implementable form.
- Object-oriented systems can be easily upgraded from small to large systems.
- Message passing techniques for communication between objects make the interface descriptions with external systems much simpler.
- Software complexity can be easily managed.
- Dynamic binding increases flexibility by permitting the addition of a new class of objects without having to modify the existing code.
- Code reuse is possible in conventional languages as well, but Object Oriented languages greatly enhance the possibility of reuse.
- Object Orientation provides many other advantages in the production and maintenance of software; high degree of code sharing.

Accessing data members and member functions:

The data members and member functions of class can be accessed using the dot('.') operator with the object.

For example, if the name of object is obj and you want to access the member function with the name printName() then you will have to write obj.printName() .

Accessing Data Members

- The public data members are also accessed in the same way given however the private data members are not allowed to be accessed directly by the object. Accessing a data member depends solely on the access control of that data member.
- This access control is given by **Access modifiers in C++.**
- There are three access modifiers : **public, private and protected.**

Example program: Accessing Of Data Members

```
#include <bits/stdc++.h>
using namespace std;
class Geeks
{
        // Access specifier
```

```
        public:

         // Data Members
        string geekname;

       // Member Functions()
        void printname()
        {
         cout << "Geekname is: " << geekname;
        }
};

int main() {

       // Declare an object of class geeks
         Geeks obj1;

        // accessing data member
         obj1.geekname = "Abhi";

        // accessing member function
        obj1.printname();
         return 0;
}
```

Output:

Geekname is: Abhi

Member Functions in Classes

There are 2 ways to define a member function:

- **Inside class definition (inline function)**
- **Outside class definition**

To define a member function outside the class definition we have to use the scope resolution :: operator along with class name and function name.

Inside the class - Method Definition

In this way function prototype is declared within class body and function is defined outside the class with the help of Scope Resolution operator (::).

Methods of class often define in public section.

```
class name
{
        private:
        ….. Data member;
public:
return-type method_name (paramete list)
        {
                …….. Function Body
        }
};
```

Outside the class - Method Definition

Syntax for defining a member function outside the class definition.

```
return-type class_name :: method_name (Parameter list)
{
        …….. Function body
}
```

Here :: pronounced as 'Scope Resolution Operator'

Use of :: Operator

```
        int x,y;
        class A
{
        public:
        int x;
void f(int x)
    {
         x= i;
        ::x= i;
         y= i;
     }
};
```

Declaration outside class

```
#include <bits/stdc++.h>
using namespace std;
class Geeks
{
        public:
        string geekname;
        int id;

 // printname is not defined inside class definition
 void printname();

 // printid is defined inside class definition
 void printid()
 {
        cout << "Geek id is: " << id;
 }
};
```

// Definition of printname using scope resolution operator ::

```
void Geeks::printname()
{
   cout << "Geekname is: " << geekname;
}
int main()
{
        Geeks obj1;
        obj1.geekname = "xyz";
        obj1.id=15;

        // call printname()
       obj1.printname();
       cout << endl;

        // call printid()
        obj1.printid();
        return 0;
}
```

Output:

Geekname is: xyz

Geek id is: 15

Data Types in C++

Data types are means to identify the types of data and associated operations of handling it.

Data types in C++ are of two types:

1. Fundamental or Built-in data types: These data types are already known to compiler. These are the data types those are not composed of other data types.

There are following fundamental data types in C++:

(i) int data type (for integer)
(ii) char data type (for characters)
(iii) float data type (for floating point numbers)
(iv) double data type

Data Type Modifiers:

There are following four data type modifiers in C++, which may be used to modify the fundamental data types to fit various situations more precisely:

(i) Signed
(ii) Unsigned
(iii) Long
(iv) Short

Variables:

A named memory location, whose contains can be changed with in program execution is known as variable. (OR)

A variable is an identifier that denotes a storage location, which contains can be varied during program execution.

Declaration of Variables:

Syntax for variable declaration is:

datatypes variable_name1, variable_name2, variable_name3,……. ;

We can also initialize a variable at the time of declaration by using following syntax:

datatypes variable_name = value;

In C++ both the declaration and initialization of a variable can be done simultaniouly at the place where the variable is used first time this feature is known as **dynamic initialization.**

e.g., float avg;

avg = sum/count;
then above two statements can be combined in to one as follows:
float avg = sum/count;

Constant:

A named memory location, whose contains cannot be changed with in program¬ execution is known as constant. (OR)

A constant is an identifier that denotes a storage location, which contains cannot be varied during program execution.

Syntax for constant declaration is:

const datatypes constant_name = value ;
e.g., const float pi = 3,14f ;

Conditional operator (? :)

- The conditional operator (? :) is a ternary operator i.e., it require three operands.
- The general form of conditional operator is: expression1? expression2: expression3 ; Where expression1 is a logical expression , which is either true or false.
- If expression1 evaluates to true i.e., 1, then the value of whole expression is the value of expression2, otherwise, the value of the whole expression is the value of expression3. For example min = a<b?a:b;
- Here if expression (a <b)is true then the value of a will be assigned to min otherwise value of b will be assigned to min.

Functions :-

Function is a named group of programming statements which perform a specific task and return a value.

1. Built-in Functions (Library Functions) :-

The functions, which are already defined in C++ Library (in any header files) and a user can directly use these function without giving their definition is known as **built-in or library functions.**

E.g., sqrt( ), toupper( ), isdigit( ) etc.

Following are some important **Header files** and useful functions within them :

stdio.h (standard I/O function)	gets( ) , puts( )

ctype.h (character type function)	isalnum( ) , isalpha( ), isdigit ( ), islower (), isupper ( ), tolower ( ), toupper( )
string.h (string related function)	strcpy ( ), strcat ( ), strlen( ), strcmp( ) , strcmpi( ) , strrev( ),strupr( ) , strlwr( )
math.h (mathematical function)	fabs ( ), pow ( ), sqrt ( ), sin ( ), cos ( ), abs ( )
stdlib.h	randomize ( ), random ( ) randomize( ) :

This function provides the seed value and an algorithm to help random( ) function in generating random numbers.
The seed value may be taken from current system's time. random( ) :

This function accepts an integer parameter say x and then generates a random value between 0 to x-1.
For example : random(7) will generate numbers between 0 to 6.

randomize( ) :

This function provides the seed value and an algorithm to help random( ) function in generating random numbers. The seed value may be taken from current system's time.

random( ) :

This function accepts an integer parameter say x and then generates a random value between 0 to x-1.
For example : random(7) will generate numbers between 0 to 6.

To generate random numbers between a lower and upper limit we can use following formula random(U – L +1) + L where U and L are the Upper limit and Lower limit values between which we want to find out random values.

For example :

If we want to find random numbers between 10 to 100 then we have to write code as: random(100 -10 +1) + 10 ; // generates random number between 10 to 100 2.

User-defined function :-

The functions which are defined by user for a specific purpose is known as user-defined function. For using a user-defined function it is required, first define it and then using.

Declaration of user-defined Function:

```
Return_type function_name(List of formal parameters)
 {
        Body of the function
 }
```

Calling a Function:-

When a function is called then a list of actual parameters is supplied that should match with formal parameter list in number, type and order of arguments.

Syntax for calling a function is: function_name (list of actual parameters);

E.g.,

```
#include int addition (int a, int b)
{
        int r; r=a+b; return (r);
}
void main ( )
{
        int z ; z = addition (5,3);
        cout<< "The result is " << z;
 }
```

The result is 8

Call by Value (Passing by value) :-

- The call by value method of passing arguments to a function copies the value of actual parameters into the formal parameters, that is, the function creates its own copy of argument values and then use them, hence any chance made in the parameters in function will not reflect on actual parameters.

- The above given program is an example of call by value.

Call by Reference (Passing by Reference) :-

- The call by reference method uses a different mechanism.
- In place of passing value to the function being called , a reference to the original variable is passed .
- This means that in call by reference method, the called function does not create its own copy of original values, rather, its refers to the original values only by different names i.e., reference.
- Thus the called **function works** the original data and any changes are reflected to the **original values.**

```
// passing parameters by reference
#include void duplicate (int& a, int& b, int& c)
 {
```

```
        a*=2; b*=2; c*=2;
 }
void main ()
{
        int x=1, y=3, z=7;
        duplicate (x, y, z);
        cout <<"x="<< x <<", y="<< y <<", z="<< z;
}
```

output :

x=2, y=6, z=14

CONTINUATION OF THE BOOK.....

STUDY MATERIALS AVAILABLE

IF YOU NEED -- PG-TRB-COMPUTER INSTRUCTOR GRADE-I- (NEW SYLLABUS 2023-2024) FULL STUDY MATERIAL WITH UNIT WISE QUESTION BANK AVAILABLE.

EDUCATIONAL PSYCHOLOGY (E/M) STUDY MATERIAL AVAILABLE.

GK WITH CURRENT AFFAIRS (E/M) STUDY MATERIAL AVAILABLE.

TAMIL ELIGIBILITY TEST STUDY MATERIAL AVAILABLE.

TET / PG-TRB /TRB-POLYTECHNIC LECTURER / BEO & DEO / UG-TRB/ TANGEDCO (TNEB) / TNPSC-VAO / JSO/TNUSRB SSC / UGC-NET / CSIR / RRB STUDY MATERIALS AVAILABLE.

TRB-POLYTECHNIC LECTURER MATERIALS

ENGLISH/ MATHS/ CHEMISTRY/ PHYSICS/ EEE /ECE/ CIVIL/COMPUTER SCIENCE IT (INFORMATION TECHNOLOGY)/ EIE / ICE / MECHANICAL/MOP AVAILABLE.

SRIMAAN COACHING CENTRE-TRICHY.

TO CONTACT: 8072230063

TRB-POLYTECHNIC LECTURER-(NEW SYLLABUS) STUDY MATERIALS AVAILABLE

- MATHEMATICS STUDY MATERIAL with Question Bank.
- ENGLISH STUDY MATERIAL with Question Bank.
- PHYSICS STUDY MATERIAL with Question Bank.
- CHEMISTRY STUDY MATERIAL with Question Bank.
- MODERN OFFICE PRACTICE STUDY MATERIAL with Q.B.
- COMPUTER SCIENCE STUDY MATERIAL with Question Bank.
- INFORMATION TECHNOLOGY STUDY MATERIAL with Q.Bank.
- ECE STUDY MATERIAL with Question Bank.
- EEE STUDY MATERIAL With Question Bank.
- MECHANICAL STUDY MATERIAL With Question Bank.
- CIVIL STUDY MATERIAL With Question Bank.
- EIE STUDY MATERIAL with Question Bank.
- ICE STUDY MATERIAL with Question Bank.

UG-TRB MATERIALS

- UG TRB: TAMIL MATERIAL WITH QUESTION BANK.
- UG TRB: ENGLISH STUDY MATERIAL +Q. BANK (T/M)

SRIMAAN COACHING CENTRE-TRICHY- TET/PG-TRB / BEO/ DEO/TRB-POLYTECHNIC/COM.INS/TNEB/UG-TRB/SCERT/JSO STUDY MATERIALS AVAILABLE- CONTACT:8072230063.

2023-24
SRIMAAN

- UG-TRB: MATHEMATICS MATERIAL WITH Q. BANK (E/M)
- UG TRB: PHYSICS MATERIAL WITH QUESTION BANK (E/M)
- UG TRB: CHEMISTRY MATERIAL + QUESTION BANK (E/M)
- UG TRB: HISTORY MATERIAL + Q.BANK (T/M & E/M)
- UG TRB: ZOOLOGY MATERIAL + QUESTION BANK (E/M)
- UG TRB: BOTANY MATERIAL +QUESTION BANK (T/M& E/M)
- UG TRB: GEOGRAPHY STUDY MATERIAL (E/M)

SCERT/DIET/GTTI (LECTURER) STUDY MATERIAL AVAILABLE.

PG-TRB MATERIALS

- PG TRB: ENGLISH MATERIAL WITH QUESTION BANK.
- PG TRB: TAMIL STUDY MATERIAL +QUESTION BANK (T/M)
- PG-TRB: MATHEMATICS MATERIAL WITH Q.BANK (E/M)
- PG TRB: PHYSICS MATERIAL WITH QUESTION BANK (E/M)
- PG TRB: CHEMISTRY MATERIAL + QUESTION BANK (E/M)
- PG TRB: COMMERCE MATERIAL WITH Q.BANK (T/M)&(E/M)
- PG TRB:ECONOMICS MATERIAL+Q. BANK (T/M & E/M)

SRIMAAN COACHING CENTRE-TRICHY- TET/PG-TRB / BEO/ DEO/TRB-POLYTECHNIC/COM.INS/TNEB/UG-TRB/SCERT/JSO STUDY MATERIALS AVAILABLE- CONTACT:8072230063.

2023-24 SRIMAAN

- **PG TRB: HISTORY MATERIAL + Q. BANK (T/M & E/M)**
- **PG TRB: ZOOLOGY MATERIAL + QUESTION BANK (E/M)**
- **PG TRB: BOTANY MATERIAL +QUESTION BANK (T/M& E/M)**
- **PG TRB: GEOGRAPHY STUDY MATERIAL (E/M)**

TNPSC-JSO STUDY MATERIAL AVAILABLE.

PG-TRB: COMPUTER INSTRUCTOR-GRADE-I (NEW SYLLABUS)-2023-2024 STUDY MATERIAL AVAILABLE

TNPSC-GROUP-I C

TNPSC-DEO (District Educational Officer(Group – I C Services) (TAMIL & ENGLISH MEDIUM) STUDY MATERIAL AVAILABLE.

TRB-BEO (Block Educational Officer) (TAMIL & ENGLISH MEDIUM) STUDY MATERIAL AVAILABLE.

10% Discount for all materials. Materials are sending through COURIER.

TO CONTACT

8072230063

SRIMAAN