I B.Sc.(CS)/I B.C.A
C++ Programming Language

UNIT-IV

2 Marks:

1. **What is File?**
   File is the collection of related records stored in a disk.
2. **What are the types of files?**
   There are three types :
   - Sequential File
   - Random File
   - Indexed Sequential file
3. **What is sequential file?**
   The records of the sequential files can be accessed one by one or sequentially.
4. **What is indexed Sequential file?**
   The records of the index sequential file can be access randomly as well as sequentially .
5. **What is the name of Header file supports file classes?**

The header file <fstream.h> contains all file supporting classes such as

```
ifstream in  //  input
ofstream out // output
fstream io     // input and output
```

6. **What is Random Access?**
   Random Access is the method of accessing a record from the file in any order.
7. **What are the functions used for random accessing?**
   The random accessing can be done using the functions
   - Seekg()
   - Seekp()
8. **What is the purpose of seekg() function?**
   The seekg() function is used to move get pointer(input) to a specified location.

Example:

Infile.seekg(10);

It moves the file pointer to the byte 10.

9. **What is the purpose of seekp() function?**
   The seekp() is used to move the put pointer(output) to a specified location.
   Example:
   Outfile.seekp(m) where m is the bytes.

1. . **What is Virtual Function?**

- ✓ A Virtual function is a member function that is declared within a base class and redefined by a derived class. To create a virtual function , the function declaration should be preceded with a keyword Virtual.
- ✓ A virtual function can be called just like any other member function . The more interesting feature of virtual function is , it is capable of supporting run-time polymorphism.
- ✓ Example:
  Virtual void fun()

```
{
      function  statements;
 }
```

- ✓ Rules for Virtual function:
  1. The virtual function must be members of some class
  2. They cannot be static members
  3. A virtual function can be a friend of another class
  4. They are accesses by using object pointers
  5. If a virtual function is defined in the base class, it need not be necessarily redefined in the derived class.

2. **What is Exception Handling?**

- ✓ Exception Handing is a built-in mechanism to handle error during run time. It si more useful to manage and respond run time errors.
- ✓ The exception handling is built upon three keywords
  - a. Try
  - b. Catch
  - c. Throw

```
Syntax:-
Try
      {
      //  try block
          }
 catch(type1 arg)
{
          // catch blcok
          }
 catch(type2 arg)
      `{
     // catch block
 }
.
.
.catch(typen arg)
       {
  //  catch block
 }
```

3. **Define stream in c++?**

- The I/O systems supplies an interface to the programmer that is independent of the actual device accessed.
- This interface is known as stream.
- A stream is a sequence of bytes.

  i)input stream- the source stream that provide data to the program

  ii)output stream- the destination stream that receives output from the program

13.Define c++ stream classes?

- C++ contains a hierarchy of classes that are used to define various streams.
- These classes are called stream classes
- Its declared inthe header file"iostream in all the programs.

1. **Define put() and get() functions?**

   The stream class define two member functions to handle I/O operations.

   i)Get()
   ii)Put()

   There are two types of get() functions

   Get(char*) – assigns input character to its argument
   Get(void) – returns input character

2. **What is the use of Getline() and write () functions?**

   Getline() – reads a whole line of text that ends with a new line character.

   Cin.getline(line,size);
   Cin.getline(name,20);

   Write() – displays an entire line

   Cout.write(line,size)
   Cout(string1,10);

3. **Define width() fiunction?**

   It used to define the width of a field for the output of an item.
   It can specify the field width for only one item.

   Cout.width(w);
   Cout.width(5);

   W – field width(number of columns)

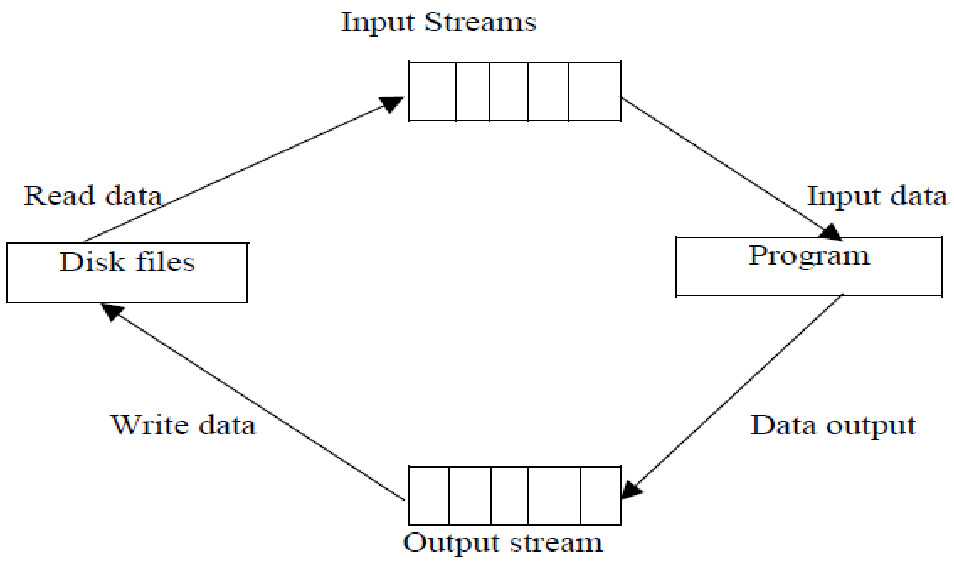
4. **What is template?**
   - It can be used to create a family of classes and functions.
   - Template is defined within parameter.
   - Templates are sometimes called parameterized classes or functions.

   ```
   Template<class T>
   Calss classname
   {
   ……….
   };
   ```

5. **Write the steps for file operations?**
   1. Suitable name of file
   2. Data type and structure
   3. Purpose
   4. Opening Method

<u>5 Marks:</u>

1. **Explain file stream classes?**
   - The console input and output operations uses file streams as an interface between the programs and files.
   - The stream that supplies data to the program is called input stream
   - The one that receives data from the program is called output stream.



- C++ contains a set of classes that defines the file handling methods.
- These include ifstream, ofstream and fstream.
- These classes are derived from fstreambase and form the corresponding iostream class.
- These classes ,designed to manage the disk files are declared in fstream
- This file is included in any program that uses files.

2. **Explain File Pointers and Manipulators?**
   - Each file has two pointers known as file pointers,
   - one is called the input pointer and the other is called output pointer.
   - The input pointer is used for reading the contents of of a given file location
   - The output pointer is used for writing to a given file location

✓ **Default actions:**

- When a file is opened in read-only mode,the input pointer is automatically set at the beginning to read from the start
- when a file is opened in write-only mode the existing contents are deleted

✓ **Functions for Manipulations of File pointer:**

seekg() Moves get pointer (input)to a specified location.
seekp() Moves put pointer (output) to a specified location.
tellg() Give the current position of the get pointer.
tellp() Give the current position of the put pointer.

✓ **Specifying the Offset:**

'Seek' functions **seekg()** and **seekp()** can also be used with two arguments as follows:

seekg (offset,refposition);
seekp (offset,refposition);

three constants defined in the ios class:

• **ios::beg** Start of file
• **ios::cur** Current position of the pointer
• **ios::end** End of file

3. **Explain about Sequential Input and Output Operations:**
   - The file stream classes support a number of member functions for performing the input and output operations on files.
   - One pairs of functions,**put()** and **get()** are designed for handling a single character at a time.
   - Another pair of functions, **write(),read()** are designed to write and read blocks of binary data.

   a) **put() and get() Functions**:

   -The function **put()** writes a single character to the associated stream.

   -The function **get()** reads a single character from the associated stream.

   b) **write() and read () functions:**
   - The functions **write()** and **read()** handle the data in binary form.
   - An int character takes two bytes to store its value in the binary form,irrespective of its size
   - The binary input and output functions takes the following form:

     infile.read ((char *) & V,sizeof (V));

     outfile.write ((char *) & V ,sizeof (V));
   - These functions take two arguments.
   - The first is the address of the variable V
   - The second is the length of that variable in bytes.
   - The address of the variable must be cast to type char*(i.e pointer to character type).

```
#include <iostream.h>
#include <fstream.h>
#include<string.h>
int main()
{
char string[80];
cout<<”enter a string \n”;
cin>>string;
int len =strlen(string);
fstream file;
file.open(“TEXT”. Ios::in | ios::out);
for (int i=o;i<len;i++)
file.put(string[i]);
file .seekg(0);
char ch;
while(file)
{
file.get(ch);
cout<<ch;
}
return 0;
```

}

4. Discuss about Error Handling during File Operations:

There are many problems encounterd while dealing with files like

- a file which we are attempting to open for reading does not exist.
- The file name used for a new file may already exist.
- We are attempting an invalid operation such as reading past the end of file.
- There may not be any space in the disk for storing more data.
- We may use invalid file name.

The function **clear()** resets the error state so that further operations can be attempted

Error Handling Functions

Function	**Return value and meaning**
eof()	Returns true(non zero value) if end of file is encountered while reading otherwise returns false(zero).
fail()	Returns true when an input or output operation has failed .
bad()	Returns true if an invalid operation is attempted or any unrecoverable error has occured.However,if it is false,it may be possible to recover from any other error reported and continues operation.
good()	Returns true if no error has occured.This means all the above functions are false.For instance,if file.good() is true.all is well with the stream file and we can proceed to perform I/O operations.When it returns false,no further operations is carried out.

Example:

```
…………..
ifstream infile;
infile.open(“ABC”);
while(!infile.fail())
{
………….. (process the file)
}
if (infile.eof())
{
……………(terminate the program normally)
}
else
if (infile.bad())
{
…………….(report fatal error)
}
else
{
infile.clear(); //clear error state
……….
}
```

<u>10 Marks:</u>

1. **Explain unformatted I/O operations?**

✓ **Overloaded operators >> and<<**

- Objects cin and cout are used for input and output of data by using the overloading of >> and << operators
- The >> operator is overloaded in the istream class and << is overloaded in the ostream class.

format for reading data from keyboard: -

```
cin>>variable1>>variable2>>…………..>>variable n
```

- where variable 1 to variable n are valid C++
- the input data for this statement would be data1 data2…………..data n
- The input data are separated by white spaces
- The operator >> reads the data character by character
- For example consider the code

```
int code;
cin>> code;
```

- Suppose the following data is entered as input 42580
- the operator will read the characters upto 8 and the value 4258 is assigned to code
- general form of displaying data on the screen is

```
cout <<item1<<item2<<…………<<item n
```

✓ **put() and get() functions:-**

- The classes istream and ostream define two member functions get(),put()
- to handle the single character input/output operations
- There are two types of get() functions.

  get(char *) -- assigns the input character to its argument
  get(void) -- returns the input character.

Example

```
Char c;
cin.get( c ) //get a character from the keyboard and assigns it to c
while( c!='\n')
{
cout<< c; //display the character on screen
cin.get( c ) //get another character
}
```

- this code reads and display a line of text.

- The operator >> can be used to read a character but it will skip the white spaces and newline character
- The function put(), a member of ostream class can be used to output a line of text, character by character.

For example

cout.put('x'); -- displays the character x

cout.put(ch); -- displays the value of variable ch.

cout.put(68); -- displays the character D.This statement will convert the numeric value 68 to a char value and displays character whose ASCII value is 68.

Example for get() and put():-

```
#include <iostream>
using namespace std;
int main()
{
 int count=0;
char c;
cout<<"INPUT TEXT \n";
cin.get( c );
while ( c 1='\n' )
{ cout.put( c);
count++;
cin.get( c );
}
 cout<< "\n Number of characters =" <<count <<"\n";
return 0;
}
Input
Object oriented programming
Output
Object oriented programming
Number of characters=27
```

✓ **getline() and write() functions:-**

- The ***getline() function*** reads a whole line of text that ends with a newline character.

  cin.getline(line,size);

- This function call invokes the function getline() which reads character input into the variable line.
- .For example consider the following code:

  char name[20];

  cin.getline(name,20);

- Assume that we have given the following input through key board:

  Bjarne Stroustrup

- This input will be read correctly and assigned to the character array name.
- Let us suppose the input is as follows:
  Object Oriented Programming
- In this case ,the input will be terminated after reading the following 19 characters
  Object Oriented Pro
- The ***write() function*** displays an entire line and has the following form:
  cout.write(line,size)
- The first argument line represents the name of the string to be displayed
- the second argument size indicates the number of characters

Example for getline and write():-

```
# include <iostream.h>
# include <conio.h>
# include <string.h>
main()
{
   char str[20];
   clrscr();
   cout<<"PLEASE ENTER THE STRING:->";
   cin.getline(str,20);
   int n=strlen(str);
   for (int i=0;i<=n;i++)
   {
      cout.write(str,i);
      cout<<"\n";
   }
   for(i=n;i>0;i--)
   {
      cout.write(str,i);
      cout<<"\n";
   }
}
```

2. Describe about all the file related functions used in C++ Language.

- ✓ To perform file I/O operation. We must include a header file <fstream.h> which contains all classes supports file operations.
- ✓ Ifstream class: (input file operation)
  This calss provides input operations or methods which contains open(), getline(), get(), read(), tellg(), seekg().
  Example :
  Ifstrem f1;
  Fstream infile;
  f1.open(“employee.dat”);
- ✓ ofstream class: (output file operation)
  This class provides output operations it supports the following methods.
  Open( )   tell( )   Put( )   write( ) Seekp( )
  Example :

Ofstream outfile;
Outfile open (“student-dat”);

- ✓ fstream class:

  It provides all function supports I/O operation.

File operation:

1. Opening the file
2. Checking the file
3. Reading / writing the file
4. Processing the records
5. Closing the records
6. Closing the file.

- ➢ Opening the file:

  File can be opened using the open() method.

  Syntax

  Filestream-class object name;
  Object-name.open(“File name”);

  Example

  Ofstream f1;
  f1.open(“student.Dat”);

- ➢ Checking the file:

  The file is checked using a method called eof().
  It returns true, if the end of file is encountered. Otherwise it returns zero.

  Example

  ```
  Ifstream f1;
  f1.open(“numbers.data”);
  f1.eof()
  while (!f1.eof())
  {
      file processing;
  }
  ```

- ➢ Writing a file: -

  The data can be written in a file using.
  File.object<<data1<<end1<<data2<<end1<<data3<<endl;

  Example :

  ofstream f1;
  f1.open(“student.Dat”);
  f1<<a1<<endl<<a2<<endl<<a3<<endl;

- ➢ Reading file

  The data can be read from the file using
  File object>>data1>>data2>>data3;

  Example

  f1>>a1>>a2;

- ➢ Processing the records:

  Records can be processed by using any mathematical and logical operations.

- Closing the file:
  File object.close();

3. **Illustrate some of an File Handling Problems.**
   a. **Write a program to create file number dat on disk to store 10 records which contains tow numeric numbers**.

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main()
  {
    ofstream f1;
    f1.open("number.dat");

     int i;
     int a1;
     int a2;
    for(i=1;i<=5;i++)
     {
       cout<<"Give Data1";
       cin>>a1;
       cout<<"Give Data2:";
       cin>>a2;
      f1<<a1<<endl<<a2<<endl;
     }
       f1.close();
       getch();
    }
```

   b. **Write a program to create a file called "student.dat" to store rno, name, m1, m2, m3, m4,m5 for ten students.**

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main( )
{
ofstream f1
f1.open("student.dat");
int rno;
char name[15];
int m1, m2, m3, m4, m5;
for (i=1; i<=10; i++)
{
cout<<"Give rno<<"Give name"<<endl;
cout<<"Give marks" <<endl;
cin>>rno>>name>>endl;
cin>>m1>>m2>>m3>>m4>>m5;
f1<<m1 <<m2<<m3<<m4<<m5<<endl;
}
f1.close();
}
```

c. Write a program to read a file “number.dat” which contains records of two numbers and find the sum.

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main()
  {
     clrscr();
    ifstream f1;
 f1.open("number.dat");
     int a1;
     int a2;
     int c1;
    while (!f1.eof( ))
     {
       f1>>a1>>a2;
       c1=a1+a2;
       cout<<"The output:"<<a1<<"  "<<a2<<c1<<endl;
       }
       f1.close();
       getch();
       }
```

d. Write a program to create a file “payroll.dat” to store 10 employees informations such as eno, ename, basicpay, da, & hra.

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main()
  {
    clrscr();
    ofstream f1;
    f1.open("payroll.dat");
    int eno;
    char nam[10];
    float basic;
    float da;
    float hra;
    int i;
    for(i=1;i<=10;i++)
     {
       cout<<"Give emp NO";
       cin>>eno;
       cout<<"Give Name:";
       cin>>nam;
       cout<<"Give Baic [ay:";
       cin>>basic;
       cout<<"Give Da:";
       cin>>da;
       cout<<"Give Hra:";
       cin>>hra;
       f1<<eno<<endl<<nam<<endl<<basic<<endl<<da<<endl<<hra<<endl;
       }
       f1.close();
       getch();
      }
```

e. Write a program to read the “payroll.dat” file to find gross pay for each employee.

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main()
  {
    clrscr();
    ifstream f1;
    f1.open("payroll.dat");
    int eno;
    char nam[10];
    float basic;
    float da;
    float hra;
float gross
   while(!f1.eof())
     {
       f1>>eno>>nam>>basic>>da>>hra;
gross=basic+da+hra
       cout<<endl;
       cout<<"Employee No:"<<eno<<endl;
       cout<<"Employee Name:"<<nam<<endl;
       cout<<"Basic Pay    :"<<basic<<endl;
       cout<<"Hra          :"<<hra<<endl;
       cout<<"DA           :"<<da<<endl;
       cout<<"Gross pay          :"<<gross<<endl;
       }
       f1.close();
       getch();
      }
```

4. **Discuss about Unformatted Binary I/O**.

Unformatted Binary I/O:

- ✓ The binary data format will be known as unformed binary data stream if we store unformatted binary data on disk it consumer very less disk memory.
- ✓ If we want to write records in Binary format we have to open the file in binary mode by using the I/O operations read and write.
- ✓ **Read ():-**

  This function is used to read record in a binary form from a file.

  Syntax :

  Ifstream read(char * buf, streamsize num);

  The read() reads num characters from the stream and put them in the buffer pointed to by buf.
- ✓ **Write( ):-**

  This function is used to write the Binary form data on file.

  Syntax :

  Ofstream write((char*buf, stream size num).

- ➢ Program to create a binary file to store ten records of students and read them .

```
#include<iostream.h>
#include<fstream.h>
#include<conio.h>
void main()
  {
    clrscr();
    struct student
            {  int rno;
               char nam[15];
               int age;
               float ht;
   };
    struct student stud;
    ofstream f1;
    f1.open("student.dat",ios::binary);
    for(int i=1;i<=5;i++)
     {
            cout<<"Give Roll No:";
            cin>>stud.rno;
            cout<<"Give Name   :";
            cin>>stud.nam;
            cout<<"Give Age    :";
            cin>>stud.age;
            cout<<"Give Ht     :";
            cin>>stud.ht;
            f1.write((char *) &stud,sizeof(struct student));
}
            f1.close();
```

```
        }
```

5.Binary File Reading:

```
        include<iostream.h>
        #include<fstream.h>
        #include<conio.h>
        void main()
          {
            clrscr();
            struct student
        {
                int rno;
                  char nam[15];
                  int age;
                  float ht;
};
            struct student stud;

             int sum=0;
             float aage;
            ifstream f1;
          f1.open("student.dat",ios::binary);
            while(!f1.eof())
            {
                f1.read((char*) &stud,sizeof(struct student));
                cout<<"The Roll No:"<<stud.rno<<endl;
                cout<<"The Name   :"<<stud.nam<<endl;
                cout<<"The Age    :"<<stud.age<<endl;
                cout<<"The Height :"<<stud.ht<<endl<<endl;
                sum=sum+stud.age;
        }
            f1.close();
            aage=sum/10;
            cout<<"The Average Age:"<<aage;
            getch();
            }
```

6. Explain Formatted Console I/O Operations:-

- C++ supports a number of features that could be used for formatting the output.
- These features include:
  - --ios class function and flags.
  - --manipulators.
  - --User-defined output functions.

ios format functions:-

Function	**Task**
Width()	To specify the required field size for displaying an outputvalue

Precision()	To specify the number of digits to be displayed after thedecimal point of float value
Fill()	To specify a character that is used to fill the unused portion ofa field
Setf()	To specify format flags that can control the form of output display
Unsetf()	To clear the flags specified

- Manipulators are special functions that can be included in the I/O statements to alter the format parameter of stream

Manipulators	**Equivalent ios function**
setw()	width()
setprecision()	precision()
setfill()	fill()
setiosflags()	setf()
resetiosflags()	unsetf()

✓ **Defining Field Width:width():-**

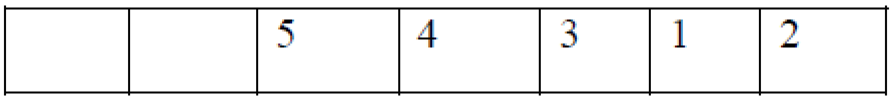
- The width() function is used to define the width of a field necessary for the output of an item.

```
cout.width(w);
```

- here w is the field width.
- The output will be printed in a field of w character wide at the right end of field.
- The width() function can specify the field width for only one item

```
cout.width(5);
cout<<543<<12<<"\n";
```

- will produce the following output:



		5	4	3	1	2

✓ **Setting Precision: precision():-**

- By using the precision ()we can specify the number of digits to be displayed after the decimal point while printing the floating point numbers.

```
cout.precision(d);
```

- where d is the number of digits

```
cout.precision(3);

cout<<sqrt(2)<<"\n";

cout<<3.14159<<"\n";

cout<<2.50032<<"\n";
```

will produce the following output:

1.141 (truncated)

1.142 ounded to nearest cent)

1.5 (no trailing zeros)

- ✓ **FILLING AND PADDING :fill()**
  - The fill() function can be used to fill the unused positions by any desired character.It is used in the following form:

    cout.fill(ch);

    Where ch represents the character
    Example:

    cout.fill('*');

    cout.width(10);

    cout<<5250<<"\n";

    The output would be:

*	*	*	*	*	*	5	2	5	0

- ✓ <u>**FORMATTING FLAGS,Bit Fields and setf():-**</u>
  - The setf() a member function of the ios class, can provide answers left justified.
  - The formatting flag specifies the format action required for the output
  - The setf() function can be used as follows:

    cout.setf(arg1.arg2)
  - arg1 is one of the formatting flags defined in the class
  - arg2,known as bit field

<u>UNIT-V</u>

<u>**2Marks:**</u>

1. **Define STL.**
   - ✓ A set of general-purpose templatized classes for data structures and algorithms that could be used as a standard approach for storing and processing of data.
   - ✓ The colletion of these generic classes and functions is called the Standard Template Library.
2. **Name the Components of STL.**
   - ✓ The STL contains several components. These omponents work in conjunction with one another to provide support to a variety of programming solutions.They are:
   - ✓ A)Containers
     B)Algorithms and
     C)Iterators

3. **Define Containers.**
   - ✓ A container is a way to store data, whether the data consists of built-in types such as int and float, or of class objects.

- ✓ The STL makes seven basic kinds of containers available, as well as three more that are derived from the basic kinds.
- ✓ Containers in the STL fall into two main categories: *sequence* and *associative*.

4. **Define Object-Oriented Analysis**.

   1.Understanding the problem.
   2.Drawing the specification of requirement of the user and the software.
   3.Identifying the objects and their attributes.
   4.Identifying the services that each object is expected to provide (interface).
   5.Establishing inter-connections (collaborations) between the objects in terms of services

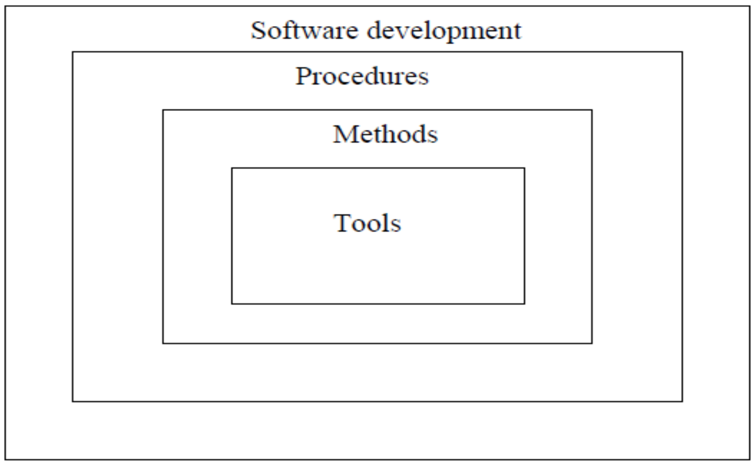
5. **Write the role of Software engineers.**

- ✓ Software engineers have been trying various tools, methods, and procedures to control the process of software development in order to build high quality software with improved productivity.
- ✓ The methods provide "how to s" for building the software while the tools provide automated or semi-automated support for the methods.
- ✓ They are used in all the stages of software development process, namely, planning, analysis, design, development and maintenance.
- ✓ The software development procedures integrate the methods and tools together and enable rational and timely development of software systems.

<u>**5Marks:**</u>

1. **Explain the concept of components of Software development**.

- ✓ Software engineers have been trying various tools, methods, and procedures to control the process of software development in order to build high quality software with improved productivity.
- ✓ The methods provide "how to s" for building the software while the tools provide automated or semi-automated support for the methods.
- ✓ They are used in all the stages of software development process, namely, planning, analysis, design, development and maintenance.
- ✓ The software development procedures integrate the methods and tools together and enable rational and timely development of software systems.



- ✓ They provide guidelines as to apply the methods and tools, how to produce the deliverables at each stage, what controls to apply, and what milestones to use to assess the progress.

- ✓ There exist a number of software development paradigms, each using a different set of methods and tools.
- ✓ The selection of particular paradigms depends on the nature of the application, the programming language used, and the controls and deliverables required.
- ✓ The development of a successful system depends not only on the use of the appropriate methods and techniques but also on the developer's commitment to the objectives of the systems.
- ✓ A successful system must:
    1. satisfy the user requirements,
    2. be easy to understand by the users and operators,
    3. be easy to operate,
    4. have a good user interface,

5. be easy to modify, Tools

6.be expandable,

7.have adequate security controls against misuse of data,

8.handle the errors and exceptions satisfactorily, and

9.Be delivered on schedule within the budget.

2. **Write a short note on Procedure-Oriented Paradigm**.

- ✓ Software development is usually characterized by a series of stages depicting the various asks involved in the development process.
- ✓ The classic life cycle is based on an underlying model, commonly referred to as the "water fall" model.
- ✓ This model attempts to break up the identifiable activities into series of actions, each of which must be completed before the next begins.
- ✓ The activities include problem definition, requirement analysis, design, coding, testing, and maintenance.
- ✓ Further refinements to this model include iteration back to the previous stages in order to incorporate any changes or missing links.
- ✓ Problem Definition:- This activity requires a precise definition of the problem in user terms. A clear statement of the problem is crucial to the success of the software. It helps not only the development but also the user to understand the problem better.
- ✓ Analysis: - This covers a detailed study of the requirements of both the user and the software. The activity is basically concerned with what of the system such as
    - o What are the inputs to the systems?
    - o What are the processes required?
    - o What are the outputs expected?
    - o What are the constraints?
- ✓ Design: - The design phase deals with various concepts of system design such as data structure, software architecture, and algorithms. This phase translates the requirements into a representation of the software.
- ✓ Coding: - coding refers to the translation of the design into machine-readable form. The more detailed the design, the easier is the coding, and better its reliability.
- ✓ Testing: - once the code is written, it should be tested rigorously for correctness of the code and results. Testing may involve the individual units and the whole systems. It requires a detailed plan as to what, when and how to test.

- ✓ Maintenance:- After the software has been installed, it may undergo some changes. This may occur due to a change in the user's requirement, a change in the operating environment, or an error in the software that has been fixed during the testing. Maintenance ensures that these changes are incorporated wherever necessary.

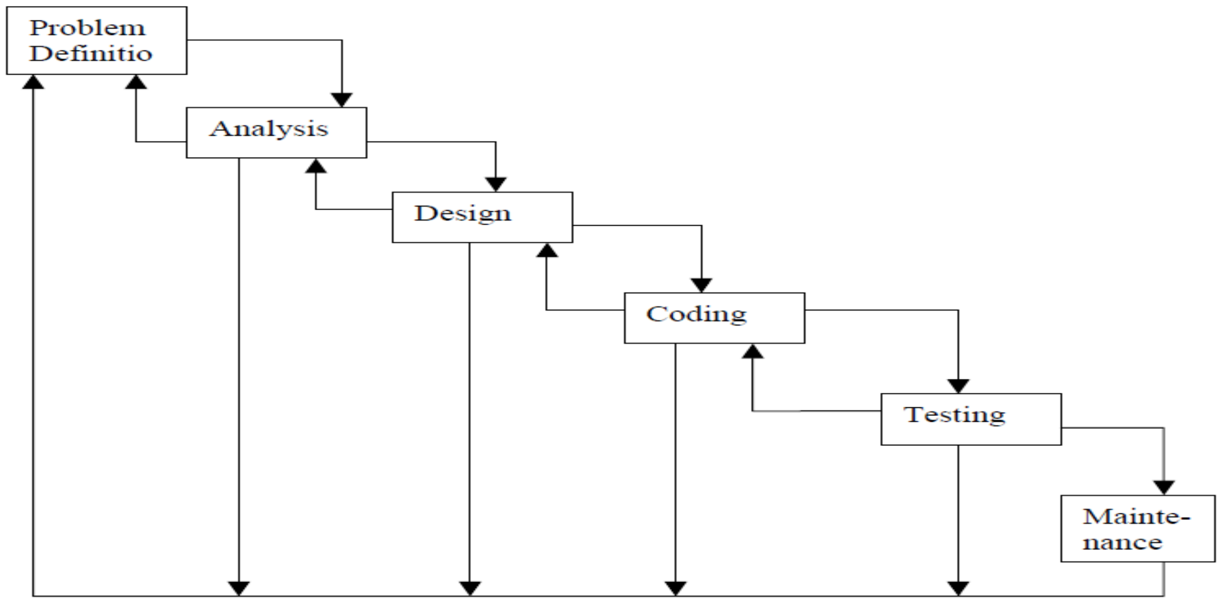


Fig. classic software development life cycle (Embedded 'water-fall' model)

3. Discuss about Procedure-Oriented Development Tools:

- The development tools may be classified as the *first generation, second generation, and third generation* tools.
- The first generation tools developed in the 1960's and 1970's are called the traditional tools.
- The second generation tools introduced in the late 1970's and early 1980's are known as the structured tools.
- The recent tools are the third generation ones evolved since late 1980's to suit the *object-oriented* analysis and design.

Process	**First generation**	**Second generation**	**Third generation**
Physical Processes	System flowchart	Context diagrams	Inheritance graphs object-relationship charts
Data Representation	layout forms grid charts	Data dictionary	Objects object dictionary
Logical Processes	Play script English narrative	Decision tables &trees Data flow diagrams	Inheritance graphs Data flow diagrams
Program Representation	Program flowcharts I/O layouts	Structure charts warnier /Orr diagrams	State change diagrams Ptech diagrams Coad/Yourdan charts

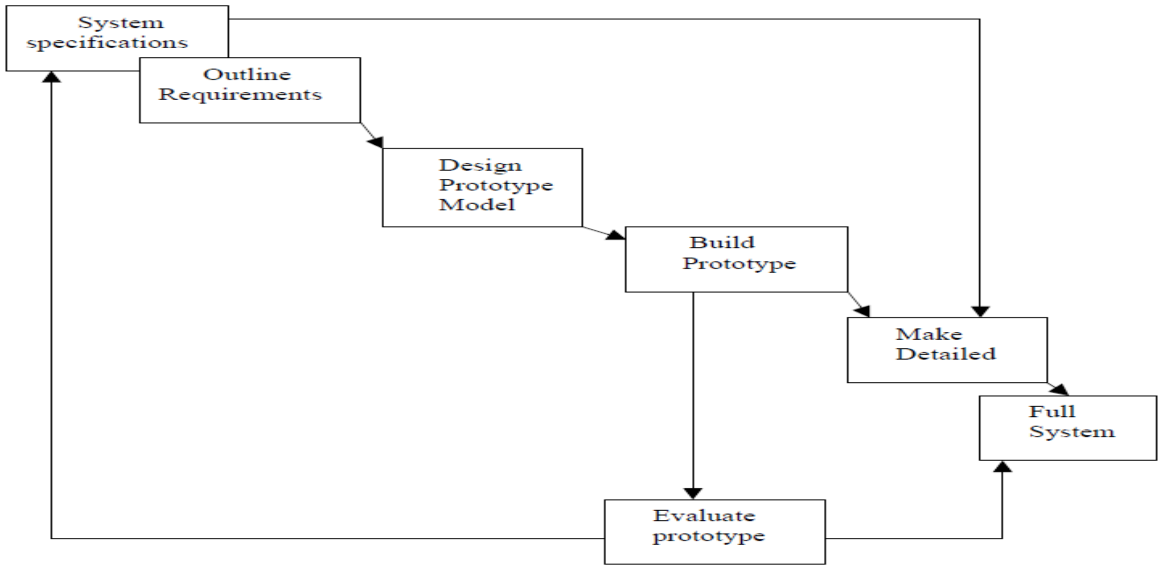
System development tools

System flowcharts: A graphical representation of the important inputs, outputs, and data flow among the key points in the system.
Program flowcharts: A graphical representation of program logic.
Play script: A narrative description of executing a procedure.
Layout forms: A format designed for putting the input data or displaying results.
Grid charts: A chart showing the relationship between different modules of a system.
Context diagrams: A diagram showing the inputs and their sources and the outputs and their destinations. A context diagram basically outlines the system boundary.
Data flow diagrams: They describe the flow of data between various components of a system. It is a network representation of the system which includes processes and data files.
Data dictionary: A structured repository of data about data. It contains a list of terms and their definitions for all the data items and stores.
Structure chart: A graphical representation of the control logic of functions (modules) representing a system.
Decision table: A table of configurations for defining a problem and the actions to be taken. It presents the logic that tells us what action to take when a given condition is true or otherwise.
Decision tree: A graphic representation of the condition and outcomes that resemble the branches of a tree.
Warnier / Orr diagrams: A horizontal hierarchy chart using nested sets of braces, psuedocodes, and logic symbols to indicate the program structure.

4. **Enumerate prototyping paradigm:**
    - ➢ To build and test a working model of the proposed system.
    - ➢ The model system popularly known as *prototype* and the process is called *prototyping*.

The benefits of using the prototype approach are:

- We can produce understandable specifications which are correct and complete as far as possible.
- The user can understand what is being offered.
- Maintenance changes that are required when a system is installed are minimized.
- Development engineers can work from a set of specifications which have been tested and approved.



<u>10 Marks:</u>

1. **Discuss about Object-Oriented Analysis**

- ✓ Object-oriented analysis (OOA) refers to the methods of specifying requirements of the software in the terms of real-world objects, their behavior, and their interactions.
- ✓ Object- oriented design (OOD), on the other hand, turns the software requirements into specifications for objects and derives class hierarchies from which the objects can be created.
- ✓ Finally, object-oriented programming (OOP) refers to the implementation of the program using objects, in an object-oriented programming language such as C++.
- ✓ By developing specifications of the objects found in the problem space, a clear and well-organized statement of the problem is actually built into application.
- ✓ These objects form a high-level layer of definitions that are written in terms of the problem space. During the refinement of the definitions and the implementation of the application objects, other objects and identified.
- ✓ All the phases in the object-oriented approach work more closely together because of the commonality of the object model. In one phase, the problem domain objects are identified, while in the next phase additional objects required for a particular solution are specified. The design process is repeated for these implementation-level objects.
- ✓ Object-oriented analysis provides us with simple, yet powerful, mechanism for identifying objects, the building block of the software to be developed. The analysis is basically concerned with the decomposition of a problem into its component parts and establishing a logical model to describe the system functions.
- ✓ The object-oriented analysis (OOA) approach consists of the following steps:
    - Understanding the problem.
    - Drawing the specification of requirement of the user and the software.
    - Identifying the objects and their attributes.
    - Identifying the services that each object is expected to provide (interface).
    - Establishing inter-connections (collaborations) between the objects in terms of services required and services rendered.

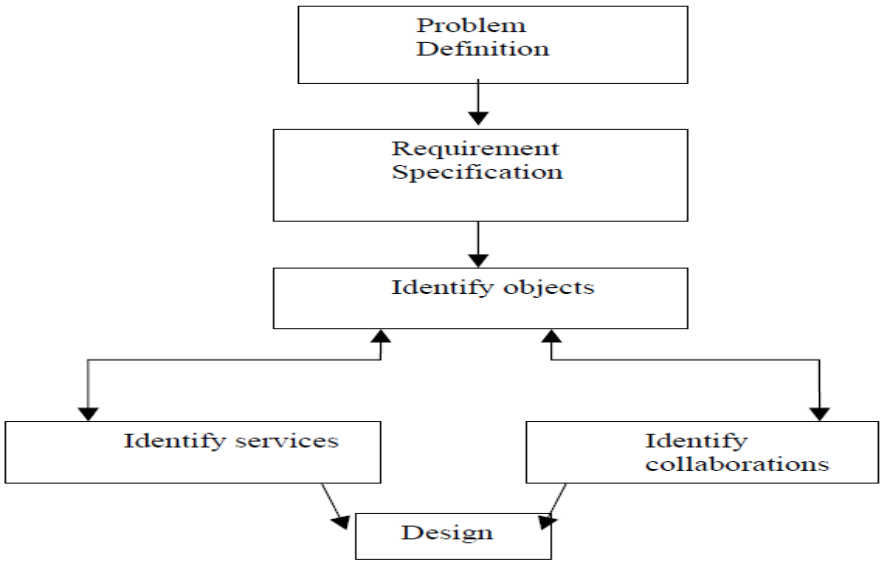


Fig. Activities of object-oriented Analyses

2. **Explain in detail about types of Containers**.

Introduction:

- ✓ A container is a way to store data,
- ✓ The STL makes seven basic kinds of containers available, as well as three more that are derived from the basic kinds.
- ✓ Containers in the STL fall into two main categories: ***sequence* and *associative*.**
- ✓ The sequencecontainers are *vector*, *list*, and *deque.*
- ✓ The associative containers are *set*, *multiset*, *map*, and *multimap*.
- ✓ In addition, several specialized containers are derived from the sequence containers.
- ✓ These are *stack*, *queue*, and *priority queue*.

Sequence Containers:

- ✓ A sequence container stores a set of elements. Each element is related to the other elements
- ✓ Each element is preceded by one specific element and followed by another.
- ✓ The STL provides the *vector* container to avoid these difficulties.This can be very time-consuming.
- ✓ The STL provides the *list* container, which is based on the idea of a linked list.
- ✓ The third sequence container is the *deque*, which can be combination of a stack and a queue. Both input and output take place on the top of the stack.
- ✓ A queue, on the other hand, uses a first-in-first-out arrangement: data goes in at the front and comes out at the back, like a line of customers in a bank.
- ✓ A deque combines these approaches so you can insert or delete data from either end. The word deque is derived from Double-Ended QUEue.
- ✓ It's a versatile mechanism that's not only useful in its own right, but can be used as the basis for stacks and queues.

Associative Containers

- ✓ An associative container is not sequential; instead it uses *keys* to access data.
- ✓ The keys, typically numbers or stings,
- ✓ There are two kinds of associative containers in the STL: *sets* and *maps*.

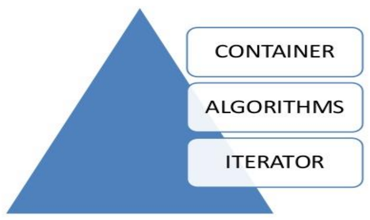
- ✓ These both store datain a structure called a *tree*, which offers fast searching, insertion, and deletion.
- ✓ Sets and mapsare thus very versatile general data structures suitable for a wide variety of applications.
- ✓ Sets are simpler and more commonly used than maps.
- ✓ A set stores a number of items whichcontain *keys*. The keys are the attributes used to order the items.
- ✓ If a set stores values of a basic type such as int, the key is the entire item stored.
- ✓ A map stores pairs of objects: a key object and a value object.
- ✓ The *map* and *set* containers allow only one key of a given value to be stored.

5. Explain the Components of Standard Template Library.

Introduction to the STL

- ✓ The STL contains several kinds of entities. The three most important are
  - i) containers
  - ii) algorithms,and
  - iii) iterators.
- ✓ A *container* is a way that stored data is organized in memory. The STL containers are implemented by template classes, so they can be easily customized to hold different kinds ofdata.
- ✓ *Algorithms* in the STL are procedures that are applied to containers to process their data in variousways.For example, there are algorithms to sort, copy, search, and merge data. Algorithmsare represented by template functions. These functions are not member functions of the containerclasses.
- ✓ *Iterators* are a generalization of the concept of pointers: they point to elements in a container.You can increment an iterator, as you can a pointer, so it points in turn to each element in acontainer. Iterators are a key part of the STL because they connect algorithms with containers.

Components of STL



Containers

- ✓ A container is a way to store data, whether the data consists of built-in types such as int and float, or of class objects.
- ✓ The STL makes seven basic kinds of containers available, as well as three more that are derived from the basic kinds.
- ✓ Containers in the STL fall into two main categories: *sequence* and *associative*.
- ✓ The sequencecontainers are *vector*, *list*, and *deque*.
- ✓ The associative containers are *set*, *multiset*, *map*, and *multimap*.
- ✓ In addition,several specialized containers are derived from the sequence containers. These are *Stack, Queue and Priority Queue.*

Algorithms

- ✓ An algorithm is a function that does something to the items in a container (or containers).
- ✓ We noted, algorithms in the STL are not member functions or even friends of container classes, as they are in earlier container libraries, but are standalone template functions.
- ✓ Suppose you create an array of type int, with data in it:

```
int arr[8] = {42, 31, 7, 80, 2, 26, 19, 75};
```

- ✓ You can then use the STL sort() algorithm to sort this array by saying

```
sort(arr, arr+8);
```

  where arr is the address of the beginning of the array, and arr+8 is the past-the-end address(one item past the end of the array).

Iterators

- ✓ Iterators are pointer-like entities that are used to access individual data items (which are usually called *elements*), in a container.
- ✓ Often they are used to move sequentially from element to element, a process called *iterating* through the container.
- ✓ You can increment iterators with the ++ operator so they point to the next element, and dereference them with the * operator to obtain the value of the element they point to.
- ✓ In the STL an iterator is represented by an object of an iterator class.
- ✓ Different classes of iterators must be used with different types of container.
- ✓ There are three major classes of iterators: forward, bidirectional, and random access.
- ✓ A *forward iterator* can only move forward through the container, one item at a time. Its ++ operator accomplishes this. It can’t move backward and it can’t be set to an arbitrary location in the middle of the container.
- ✓ A *bidirectional iterator* can move backward as well as forward, so both its ++ and -- operators are defined.
- ✓A *random access iterator*, in addition to moving backward and forward, can jump to an arbitrary location.