① exception are unusual conditions that a program may encounter while execution. when a program encounters an exception, it is important that language must have some mechanism for identifing and dealing with it.

It perform following tasks:-

ⓐ Hit exception i.e find the unusual condⁿ

ⓑ Throw the exception i.e inform that error has occured.

ⓒ Catch the exception i.e receive the error information.

ⓓ Handle the exception i.e take the action for correction problem.

In C++ following keywords are used for exception handling

(I) try

(II) Catch

(III) throw

```
try
{
  throw exception;  // It can be written inside the try block. when an exception is detected, it is thrown using throw statement which is written inside
}
Catch(arg)
{
}
```

// It catches the exception thrown using throw statement

example

```
void main()
{
 int f,s ;
 cout<<"Enter first No";
  cin>>f ;
 cout<<" Enta the second";
  cin>>s;
 try
  {
  if (s!=0)
  {
   cout<< f/s;
  }
  else
   {
    throw(s);
   }
  }
 catch(int n)
  {
  Cout<<" There is an exception division by zero";
  }
  getch();
  }
```

① f=6
s=3

② f=6
s=0] →

Multiple catch Statement

```
class sum
{
Private: int a,b,c;
Public: void getdata();
        void sum1();
       };
void sum:: getdata()
     {
      cout<<"Enter 1st no";
      cin>>a>>b;
     }
void sum:: sum1()
     {
       c=a+b;
      cout<<c;
     }
```

discussion → I/P → int
O/P → Sum of int

```
template<class T>
class sum
{
 Private: T a,b,c;
 Public:  ——

          ——

};
template<class T>
void sum<T>:: getdata()
     {
      cout<<"  ——  ";
       cin>>a>>b;
     }
void sum<T>:: sum1()
       {
        c=a+b;
        cout<<c;
       }
void main()
{
 Sum<int> S1;
 Sum<float> S2;
 S1.getdata();
 S1.Sum1();
```

General formate of template

```
template<class T>
class class_Name
{

};
```

class Template with multiple parameter →

```
template <class T1, class T2>
  class sum
  {
    T1 x;
    T2 y;
    Public: sum(T1 m, T2 n)
            {
              x=m;
              y=n;
            }
           Void add()
           { cout<< x+y;
           }
    };
  void main()
    {
    Sum < int, int > S1(10,20);
    Sum < int, float > S2(10,20.5);
    Sum < float, float > S3(10.5,10.5);
    S1. add();
    S2. add();
    S3. add();
    getch();
    }
```

function Template →

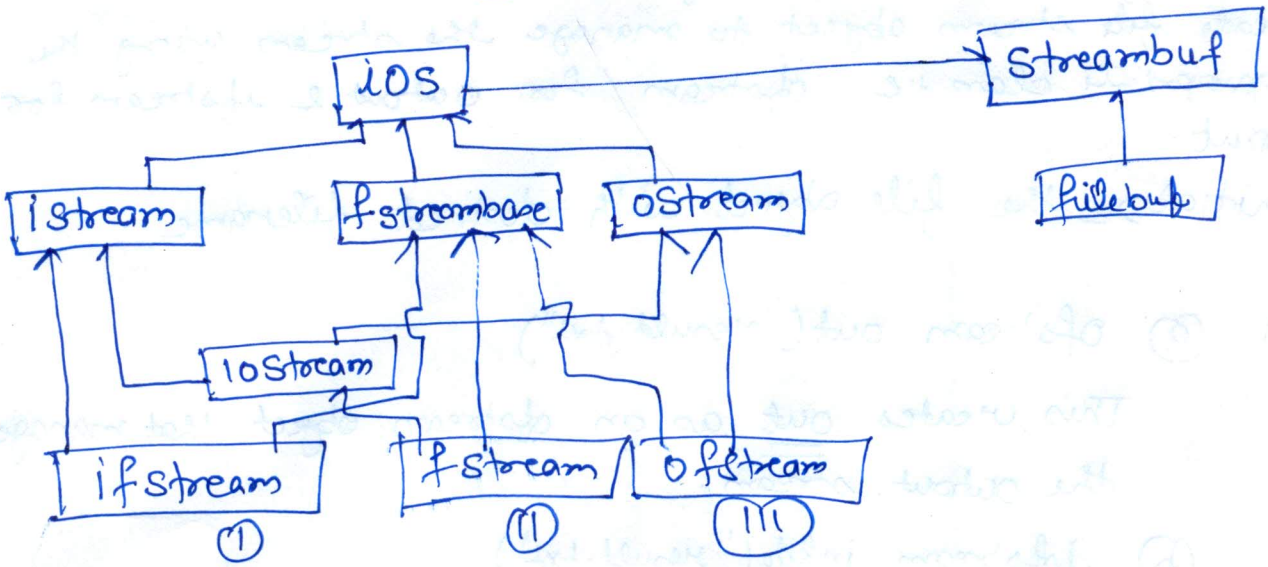
```
template < class T >
void sum( T&a, T&b )
 {
    cout<<a+b;
 }
void sum(int &a, int &b);
void sum (float &a, float &b);
void main()
 {
   int x,y;
   cin>>x>>y;
   sum(x,y);
   getch();
 }
```

```
float x1,y1;
cin>>x>>y;
sum(x1,y1);
```

function Template with multiple parameter

Member function as a Template

CH- file Stream



(a) operation with file

firstly, we will see how files are opened & closed.
A file can be defined by following class i.e

(I) if it is declared by ifstream class then that can be used for reading from file.

(II) " " " by ofstream class " "
" " writing onto file purpose.

(III) " " by fstream class " "
both reading & writing.

(1) opening a file → (1) using constructor fn of class.
(2) Using member function open() of class.

↳ It is used to use only one file in the stream

if we want to manage multiple files using one stream.

① using Constructor → ⇒A filename is used to initialize the file stream object

ⓐ Create file stream object to manage the stream using the appropriate class i.e ofstream for output & ifstream for input.

ⓑ initialize the file object with desired filename.

For eg ⓐ ofstream out("result.txt")

This creates out as an ofstream object that manage the output stream.

ⓑ ifstream infile("result.txt")

infile as an ifstream object that attaches it to result for reading

Program → ① WAP which creates a file and writes some integer.

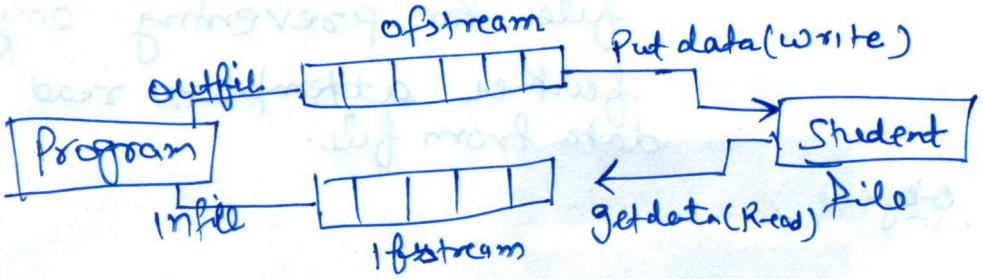
```
#include<fstream.h>
void main()
{
  ofstream out("student.txt");
  cout<<" Enter the name of student";
        char name[10];
        cin>>name;
        out<<name;        // write to file student
}
```

② WAP which read the content of student

```
void main()
{
    ifstream infile("student.txt");
    char name[30];
    infile>>name;        // read name from file
    cout<<" student Name is"<<name;

}
```

(11) Using open() function → It is used to open multiple file that use the same stream object




```
ofstream outfile("Student.txt")
ifstream infile("Student.txt")
```

fig → Two file stream working on same file

for eg →

```
void main()
{
ofstream fout;                    // output stream
fout.open("Student");
fout<<"Akash kumar";
fout<<"Anil kumar";
fout.close();
fout.open("Branch");
fout<<"CSE";
fout<<"ECE";
fout.close();
char line[80];
ifstream infile;                  // Reading stream
infile.open("Student");
cout<<"Student name is";
while(infile)
{
infile.getline(line,80);          //read line
cout<<line;                       // out display
}
infile.close();
```

EOF (end of file) → to detect the end of file for preventing any further attempt to read data from file.

while(stream object)

File Modes → open() takes two arguments.

Syntax: stream_object.open("file name", mode);

ios::app	→	Append to end of file
ios::ate	→	go to end of file on opening
ios::binary	→	open as binary file
ios::in	→	open file for reading only
ios::nocreate	→	open fails if file already exist
ios::out	→	open a file for writing
ios::trune	→	delete content of file