Title: UNIT-III

Ex.No.: 1) Exception Handling.

Date :

2) Streams & Formatted I/O.

3) File handling.

4) Namespace

5) String objects

6) Standard template Library.

1) Exception handling:-

Exception refers to unexpected condition in a program.

Error occuring in one particular part of the program are reported to another part of the program, known as the calling environment.

Traditional error handling.

1) Returning error Number. → arg is wrong ⇒ return error code
2) Global flag manipulation ↳ error variable.
3) Abnormal termination. → error would eventually crash the pgm.

Page No.

Need for exception handling:-

→ Dividing the error handling.

→ To solve the obj distroy pbm.

→ Separating error reporting and error handling.

→ To provide unconditional termination & programmer prefered termination.

Components of Exception handling Mechanism:-

1) try - for indicating pgm area where the exception can be thrown.

2) throw - for throwing an exeption.

3) catch - for taking an action for Specific exeption.

Try

```
try
{
    //code for raising exception
}
```

Catch.

obj name, nameless obj

```
catch(e)
{
    Action for handling
      an exception
}
```

Page No.

Title:

Ex.No.: Date :

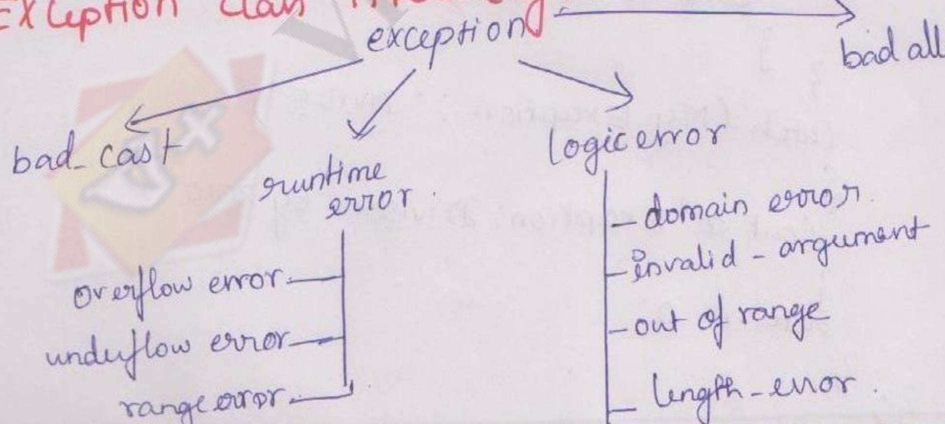
Throw

Throw e → obj name(or) nameless obj(or)
keyword default, Nothing.

Error handling code must perform the following task.

1. Detect the pbm causing exception (Hit exception)
2. Inform that an error has occured (Throw an exception)
3. Receive the error information (Cath the exception)
4. Take corrective actions (Handle the exception).

Exception class Hierarchy :-



Page No.

```
#include <iostream.h>
#include <conio.h
class MyException.
{ int num;
  public:
      MyException (int a)
        { num=a;
         }
    class DIVIDE{}; // abstract class used in exception.
    int Divide(int x)
     {
       try
        {
         if (x==0)
           throw DIVIDE(); //raise exception.
          cout<<"this stmt will not be executed
        else
          {
           cout<<"Trying division succeeded";
           cout<< num/x;
          }
        }
       catch (My Exception :: DIVIDE)
       {
        Cout<<"Exception! Division by zero"
       }
       return 0;
     }
  };
```

Page No.

Title:

Ex.No.: Date :

```
void main ()
 {
  MyException ex(10);
  ex.Divide(0);
 }
```

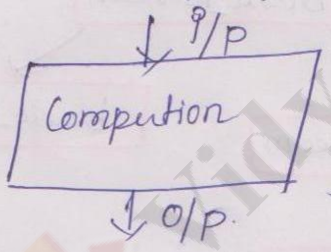
2) Streams and Formetted I/O

Stream:-

Stream like constructs used for provinding I/p & O/p.

Clasified in two types.

1. I/p stream
2) O/p stream.



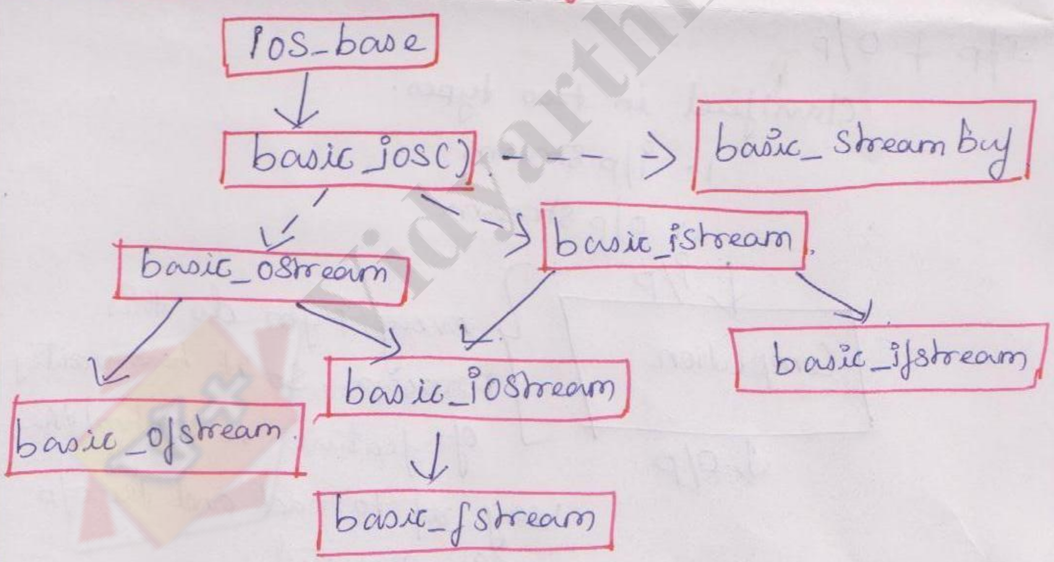
} every pgm do this process, so it has variety of feature to control the way ~~of~~ to read and the o/p is generated.

Page No.

Predefined & Wide character Streams.

Stream	Meaning	Default Source.
cin	Standard i/p stream	keyboard.
cout	standard o/p stream	screen.
cerr	Standard error stream with no buffer	,,
clog.	,, with buffer.	,,

C++ stream classes hierarchy :-



Page No.

Title:

Ex.No.: Date :

FORMATTED I/O.

ios member function	use.
width()	— The o/p will take up the width specified.
precision()	— It specifies the precision of the floating pt no. . By default it is six digit after decimal pt.
fill ()	— The character for filling up the unused portion of field.
~~Sett~~ Setf()	— format flags that controls display like left or right or justification, padding after sign.
unsetf().	→ It provides the undo operation for above mentioned operation with setf().

Page No.

File handling:-

A file is a collection of related info. commonly it represents pgm's & data.

Data → numeric, alphabetic, alphanumeric.

File → free form such as text files or may be formatted.

[File handling means the operations that we can perform on files such as file creation, file open, read, write, rewind, ~~close~~ close & delete].

File I/O involves the following steps:-

1) check for hard disk space to store file.
2) Ask for association of a physical file with the logical file.
3) Specify the type of file. 2 Type.
    1) text file → to open the text file (eg Notepad)
    2) binary file → can be read using pgm only.
4) Open the file.
5) R (or) W operation to the F's.
6) Close the File.

Page No.

Title:

Ex.No.: Date :

File stream classes :-

operations.

i) ifstream – open(), get(), getline(), read(), seekg(), tellg().
[i/p file]

2) ofstream. – open(), put(), seekp(), tellp(), write()
[o/p file]

3) fstream. – It support ifstream & ofstream operations.
[I/O file]

Text file program :- [text file]

```
#include<iostream.h>
#include <fstream.h>
void main()
{
  char i/p, o/p;
  ofstream EntryFile("Test.~~dat~~");
  cout<<"input:"<<endl;
  while(1)
  {
    cin>> input;
    if (input = ='$') break;
  } EntryFile << input;
```

Page No.

```
Entry File. close ();
cout <<" output :"<<endl;
ifstream DisplayFile ("Test.txt");
while (!. DisplayFile.eof())
 {
  DisplayFile . unsetf (ios::skipWS);
  DisplayFile >> output;
  cout << output;
 }
 DisplayFile . close();
}
Input :
    abcd $.
output :
     abcd.
```

Page No.

Title :

Ex. No. : Date : Pg : 5-23

Binary File :-

```
#include <iostream.h>
#include <fstream.h>
void main()
{
  char input, output;
  ofstream EntryFile ("Test.dat", ios::in|ios::
                                    binary); //
  Cout << "Input :" << endl;
  while (1)
  {
     Cin >>input;
      if (input == 'e') break;
      EntryFile <<input;
   }
   EntryFile.close();
   Cout<<" Output:" <<endl;
   ifStream DisplayFile ("Test.dat", ios::
                 in|ios:: binary); //binary mode
   while (!DisplayFile.eof())
```

Page No.

```
{
    DisplayFile.Unsetf(ios::skipws);
    DisplayFile>>Output;
    Cout << Output <<"\t";
  }
  DisplayFile.Close();
}
```

Input:
abcde

The output generated by this program is:

Output:

```
a    b    c d
```

Page No.

Title :

Ex. No. : Date :

Name Spaces :-

To localize the names of identifiers to avoid name collision.

In S/w projects lead to name conflict pbm, especially with multiple class libraries are in operation.

Name conflict problem

↓

It was impossible for the compiler to distinguish b/w diff classes that had identical names.

The Using Syntax :-

1) using declaration

2) using directive. → using namespace std;

```
#include<iostream.h>
void main ()
{ using std::cout;
  cout<<"welcome to namespace";
}
```

Page No.

Declaring a function inside and defining outside the Namespace

```
#include<iostream>
using namespace std;
namespace myLib.
 {
   int a,b;
   Void Get();      // declaration.
   void put();
 }
void myLib::Get()  // defining outside the namespace.
 { cin>>a>>b;
 }
void myLib::put().
  { cout<<a<<b;
  }
using namespace mylib;  // using directive
 void main ()
  {
   Get();
   put();
  }.
```

Page No.

Title :

Ex. No. :

Date :

Declaring a member function inside and defining Outside the Namespace :-

```
#include <iostream>
using namespace std;
namespace mylib
{
    class student      // class declaration
    {
        int roll;
        char name[20];
        Public :
            void Get();    //member function declar
            void Put();    //     "      "      "
    };
}
void mylib::student::Get()//defining outs
                                  the class
{
    Cin >>mylib::student::roll;  //using
    Cin >>mylib::student::name;  memb

}
```

Page No.

```
void mylib::Student::Put() //defining outside
                              the clas
{
   Cout <<mylib::Student::roll; //using data
                                     membe
   Cout <<Mylib::Student::name;
}
using namespace mylib;  //using directive
void main()
{
     mylib::Student S;    //object Creation
     S.Get();
     S.Put();
}.
```

Page No.

Title :

Ex. No. : String Object :-

Date :

1) Creating String.

1) Def a str obj in a normal way.

String S1;

2) Def a str obj using initialization

String S2.

String S1(S2) or String S1=S2.

3) Def a string obj using a constructor.

String S1("hi");

(or)

String S1="hi";

2) Substring Operator:-

i) find () eg:-

String youth ("Bill is so young, so young");

int first = youth.find ("young");

value is 11

2) at().

String S1 ("hello");

S1.at(0)

h is a char at location 0

Page No.

3) Insert ()

eg string s1("Miss summer");

s1.insert(5," ~~[illegible]~~ Ram");

o/p Miss Ram Summer.

4) erase ()

String s ("The summer-time");

S.erase (4,7) // Start position :4 Quantity 7.

o/p The time.

5) append ()

String s1 = "hello".

S1.append ("how are you?");

o/p hello how are you?.

6) Replace () String s1("There they go again!"), S2("bob & Bill");

S1.replace (6,4, S2)

o/p There bob and bill go again.

Operations involving multiple strings:-

a) + String s1 = "hello";

String s2 =" how are you?";

String S3 = S1+S2.

Page No.

Title :

Ex. No. : Date :

Swapping two strings.

```
String s1 = "one";
String s2 = "two";
s1.swap(s2);
```

Standard Template Library (STL)

> STL is a collection of generic algorithms and containers that communicate through iterators.

> STL is different from normal libraries.

STL components

1) containers
   - Sequence containers
   - Sorted Associative containers
     - Set
     - Map
2) algorithms.
3) iterators.
   - Address mechanism
   - functional mechanism
4) function objects.
5) adaptors.
6) allocators.

Page No.