UNIT - 5

Exception handling

Exception: Exception is an abnormal termination of the program which is executed in a program at runtime or it may be called at run time when the error occurs. the Exception contains warning messages such as invalid argument, insufficient memory, and divided by zero.

Exceptions provides a way to react to exceptional (abnormal) conditions in programs by transferring program control to special functions called handlers.

Exception handling is a newly added feature in c++ that is now almost supported by all compilers. The main aim of exception handling is to detect and report exceptional condition(s).

* Exception handling deals only with synchronous exceptions

** Principles of exception handling:-

c++ has a well organized, object oriented method to control runtime errors that occur in the program. The goal of Exception handling is to create a routine or method that detects and sends an exceptional condition in order to execute suitable actions.

The routine need to carry out the following responsibilities

(1) Detect the problem

(2) Warn that error has been detected

(3) Accept the error message

(4) perform the accurate actions without troubling the user.

An exception is a object. It is send from part of the program when an error occurs to the part of program that is going to control the error.

***

The keyword try, catch and throw :-

C++ Exception handling is built upon three keywords try, catch and throw.

try block:

try block containing group of statements that may generate exceptions. when an exception is generated it is thrown using the keyword "throw". when try block throws an exception the program control leaves the try block and enters the catch block. An exception is an object

transmits information about unusual circumstances. The type indicates the type of exception that catch block handles. arg is an optional parameter. if the type of exception thrown from try block matches with argument (arg) type in the catch block then catch block is executed for handling exception(s) After handling the exception handler code in the catch block the control goes to statement immediately following catch block. however, incase the type of argument (arg) does not match with type specified in catch block the program is aborted by using abort() which is invoked by default.

Syntax:

```
try
{
    statement-1;
    statement-2;
    throw exception;
}
```

** throw keyword:-

An exception is thrown using keyword throw from inside the try block. the type of exception thrown will be handled by catch block.

Syntax:

```
throw exception;
throw (exception);
throw;
```

The last form of throwing an exception is usually used for rethrowing exceptions.

The above exception operand in other two syntaxes may be "any type" and it may be constant. The catch block is associated with try block. catches the exceptions thrown. the throw keyword can be placed in function or in a nested loop but it should be in the try block. After throwing exception control passes to catch statements.

Catch block:

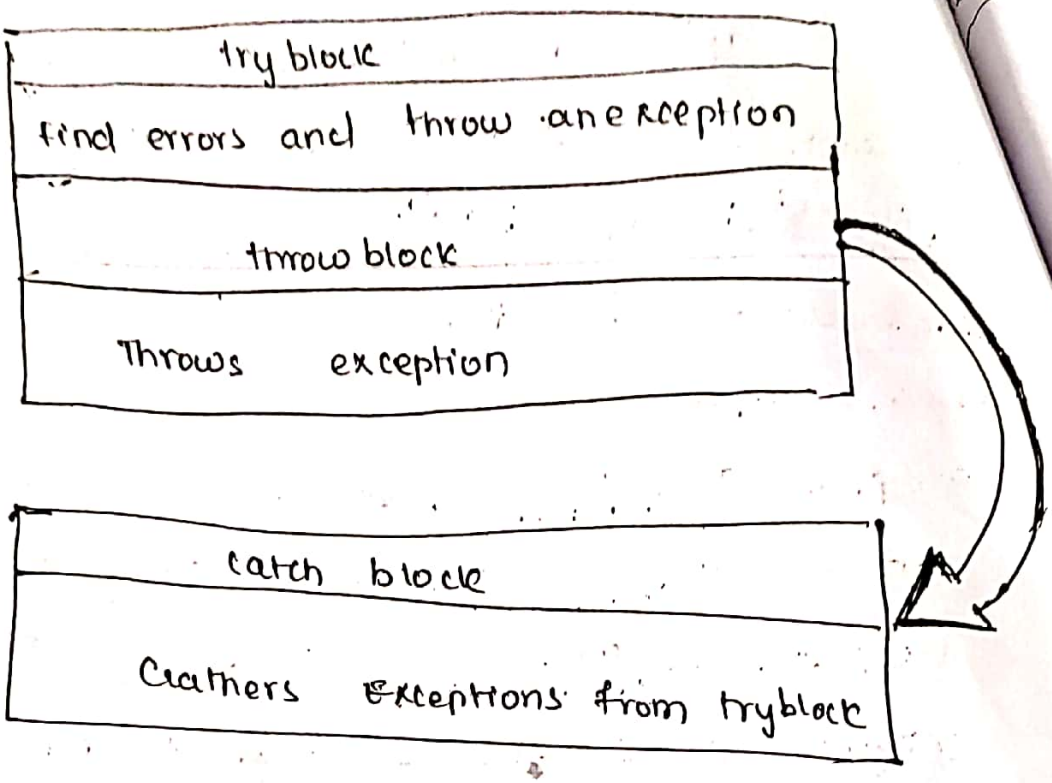
Similar to try block catch block also contains a series of statements enclosed in curly braces. It also contains an argument of an exception type in parenthesis

Syntax

```
try
{
    Statement-1;
    Statement-2;
    throw exception;
}
catch ( Type argument)
{
    Statement-1;
}
```

when an exception is found the catch block is executed. The catch statement contains an argument of exception type and it is optional. If the type of exception thrown from try block matches with argument type of catch block then catch block is executed for handling exception. In case of mismatch the program is aborted using abort() which is invoked by default.

Exception mechanism:



Ex:

Program for divide by zero

```
int main()
{
    int a = 6, b=0;
    try
    {
        if(b==0)
            throw b;
        else
            cout<<"division=" <<a/b;
    }
    catch( int b)
    {
        cout<<" Error: Divide by zero is not
                                allowed"<<endl;
    }
}
```

ultiple catch statements:

It is possible to associate more than one catch block with a single try block. This is usually done when program segment has more than one condition to throw as an exception. In such cases, when an exception is thrown the exception handler are searched to find an appropiate match. The first catch block that matches type of exception ~~ee~~ thrown is executed. After execution program control goes to first statement after the last catch block. this means other catch blocks are ~~bl~~ ignored. However if no match is found, then program is terminated using the default abort().

Syntax:

```
try
  {
    ....
    throw exception;
    ....
  }
catch (type1 arg1)
    {
      ....
    }
catch (type2 arg2)
    {
      ....
    }
  .
  .
  .
catch (typen argn)
    {
      ....
```

In case more than one catch block matches type of exception thrown, the first catch block that matches the exception is executed.

Ex: Example on array index out of bounds

```
int main()
{
    int a[5]= { 10,20,30};
    int index;
    try
    {
        cout " Enter Index",
        cin >> index;
        if (index <0)
            throw "positive value required",
        else if ( index >=5)
            throw index;
        else
        {
            a[index] = 100;
            cout<< a[index];
        }
    } // end of try block
catch( const char *e)
{
    cout<<" Index should not be negative";
}
```

Contd.

```
catch( int i)
{
cout<<" Array index out of bounds due to i="<<i<<
                                                endl;
}  // end of catch block.
}  // end of main()
```

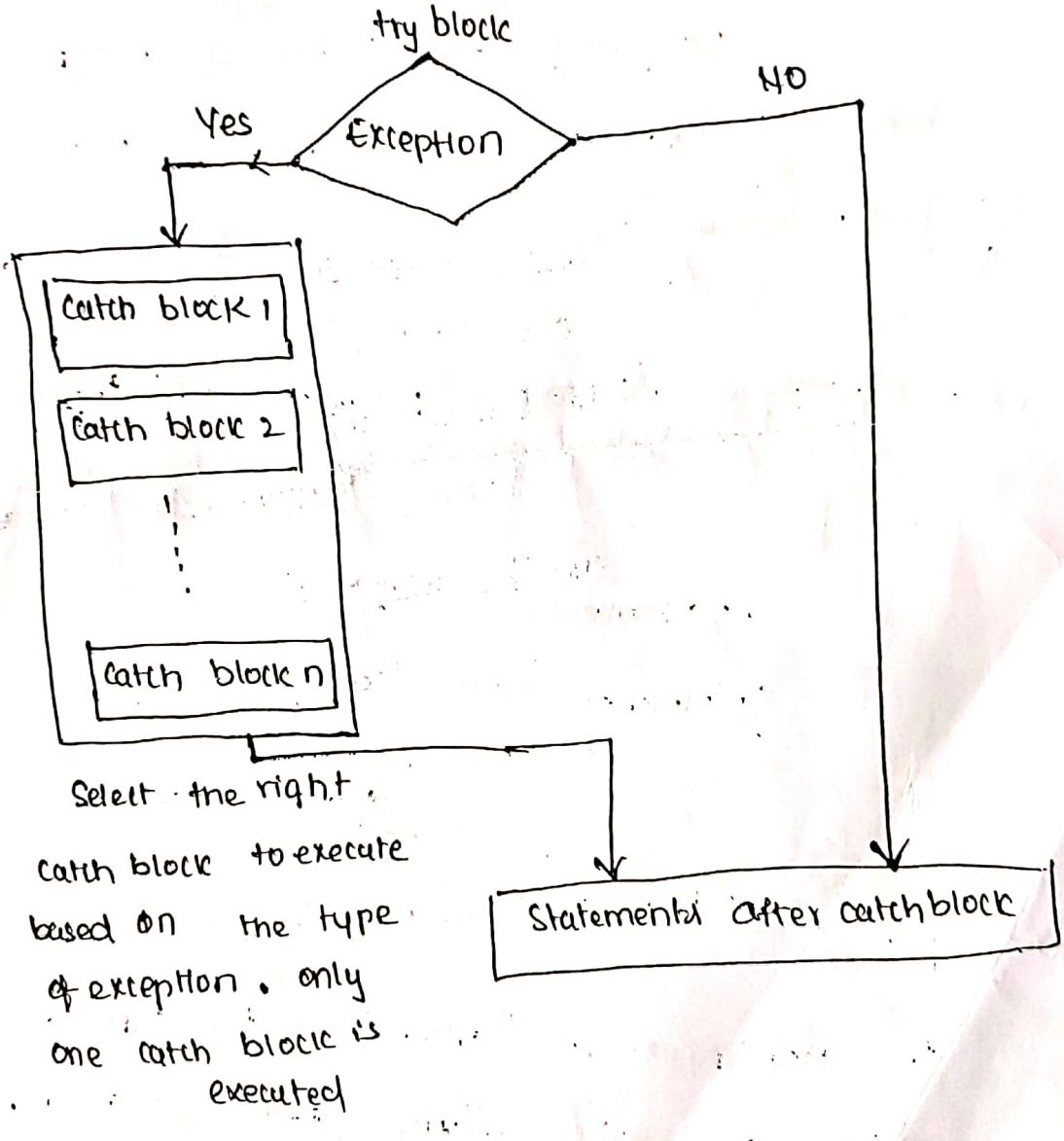


fig 9: Execution of try-catch