EXPERIMENT NO : 11

Title: Write a program to implement Exception Handling

Objectives: To handle runtime errors in programming.

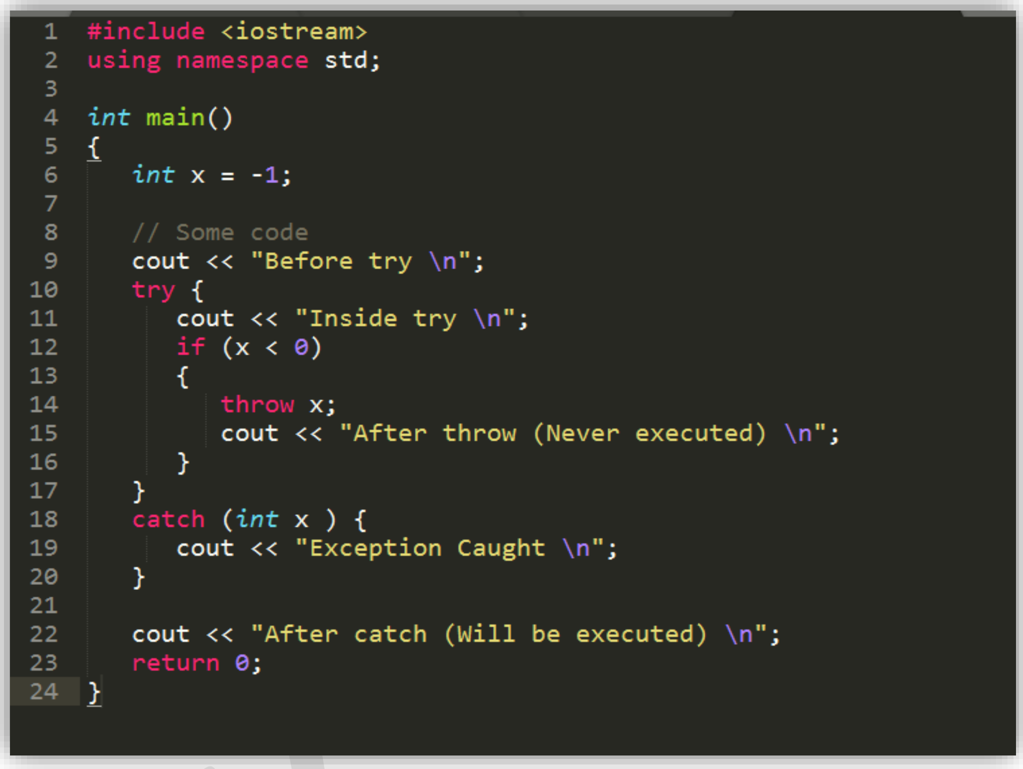
Key Concepts: Exception, try, throw, catch.

Theory: One of the advantages of C++ over C is Exception Handling. C++ provides following specialized keywords for this purpose.

try: represents a block of code that can throw an exception.
catch: represents a block of code that is executed when a particular exception is thrown.
throw: Used to throw an exception. Also used to list the exceptions that a function throws, but doesn't handle itself.
Following are main advantages of exception handling over traditional error handling.

1. Separation of Error Handling code from Normal Code: In traditional error handling codes, there are always if else conditions to handle errors. These conditions and the code to handle errors get mixed up with the normal flow. This makes the code less readable and maintainable. With try catch blocks, the code for error handling becomes separate from the normal flow.
2. Functions/Methods can handle any exceptions they choose: A function can throw many exceptions, but may choose to handle some of them. The other exceptions which are thrown, but not caught can be handled by caller. If the caller chooses not to catch them, then the exceptions are handled by caller of the caller.
   In C++, a function can specify the exceptions that it throws using the throw keyword. The caller of this function must handle the exception in some way (either by specifying it again or catching it)
3. Grouping of Error Types: In C++, both basic types and objects can be thrown as exception. We can create a hierarchy of exception objects, group exceptions in namespaces or classes, categorize them according to types

Example:



```cpp
#include <iostream>
using namespace std;

int main()
{
    int x = -1;

    // Some code
    cout << "Before try \n";
    try {
        cout << "Inside try \n";
        if (x < 0)
        {
            throw x;
            cout << "After throw (Never executed) \n";
        }
    }
    catch (int x ) {
        cout << "Exception Caught \n";
    }

    cout << "After catch (Will be executed) \n";
    return 0;
}
```

Program analysis:

A) Class Date :

Member Variables: Day, Month, Year – Declare all Variables Private

Member Function : No Argument Constructor – Date( ) & setDate( ) Function

- If entered date is 31<Day<0 and 12<Month<0 (check this condition in setDate( ) function)then throw the exception and handle the error.
- Test the program for different input cases.

B) Class Person :

Member Variables: Person_Name , Person_ID – Declare all Variables Private

Member Function : No Argument Constructor & setName( ) Function

- If entered Name is Empty/Null (check this condition in setName( ) function) then throw the exception and handle the error.
- Test the program for different input cases.