UNIVERSIDAD NACIONAL DE SAN AGUSTÍN

ASIGNATURA: PROGRAMACIÓN DE SISTEMAS

PRÁCTICA LABORATORIO N°07

OBJETIVO

- La presente práctica de laboratorio tiene como objetivo trabajar con Excepciones en C++.

TEMA

- Introducción.
- Sintaxis de una excepción.
- Ejemplo de un programa con excepciones.

MARCO TEÓRICO

INTRODUCCION:

Una excepción es un mecanismo de gestión de errores incorporado. Gestiona errores que pueden ocurrir debido a una mala entrada por parte del usuario, un mal funcionamiento en el hardware, un argumento inválido para un cálculo matemático, etc. Permite gestionar y responder a los errores en tiempo de ejecución. Las excepciones están construidas a partir de tres palabras clave: try, catch y throw. Cualquier sentencia que provoque una excepción debe haber sido ejecutada desde un bloque try o desde una función que este dentro del bloque try. Cualquier excepción debe ser capturada por una sentencia cath que sigue a la sentencia try, misma que es causante de la excepción.

SINTAXIS DE UNA EXCEPCIÓN:

```
try{
    cuerpo;
    throw;
}
catch(tipo1 arg){
    bloque catch;
}
catch(tipo2 arg){
    bloque catch;
}

catch(tipoN arg){
    bloque catch;
```

}

El lenguaje de programación C++ soporta una forma más elegante para el manejo de excepciones que su contraparte el C estándar y esta consiste en el mecanismo try, throw y catch que son las tres palabras claves de una excepción. La lógica del mecanismo mencionado consiste en:

1. Dentro de un bloque try se pretende evaluar una o más expresiones y si dentro de dicho bloque se produce un error se lanza por medio de throw la excepción, misma que deberá ser capturada por un catch específico.
2. Puesto que desde un bloque try pueden ser lanzados diferentes tipos de errores de excepción es que puede haber más de un catch para capturar a cada uno de los mismos.
3. Si desde un try se lanza una excepción y no existe el mecanismo catch para tratar dicha excepción el programa se interrumpirá abruptamente después de haber pasado por todos los catchs que se hayan definido y de no haber encontrado el adecuado.
4. Los tipos de errores lazados pueden ser de un tipo primitivo tal como: int, float, char, etc. aunque normalmente las excepciones son lanzadas por alguna clase escrita por el usuario o por una clase de las que vienen incluidas con el compilador.

En el ejemplo siguiente se muestra un programa de cómo lanzar una excepción de tipo int dentro del bloque try, y como capturar el error por medio de catch.

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

int main()
{
    try {
            throw 125;
    }
    catch (int)
    {
            cout << "Error tipo entero capturado" << endl;
    }
    cin.get();
    return 0;
}
```

Los errores pueden deberse a una multitud de situaciones muchas veces inesperadas, por tal motivo, en C++ existe una forma de manejar excepciones desconocidas (genéricas) y es buena idea que si uno como programador está escribiendo un controlador de errores incluya un catch para capturar errores inesperados. Por ejemplo, en el siguiente código se escribe un catch que tratará de capturar cualquier error inesperado.

Las excepciones son en realidad errores durante la ejecución. Si uno de esos errores se produce y no implementamos el manejo de excepciones, el programa sencillamente terminará abruptamente. Es muy probable que si hay ficheros abiertos no se guarde el contenido de los buffers, ni se cierren, además ciertos objetos no serán destruidos, y se producirán fugas de memoria.

En programas pequeños podemos prever las situaciones en que se pueden producir excepciones y evitarlos. Las excepciones más habituales son las de peticiones de memoria fallidas.

❖ EJEMPLO DE UN PROGRAMA CON EXCEPCIONES:

A continuación se presenta un ejemplo de un error que se causa en un programa que realiza divisiones entre 2 números, que al momento de pedirle el denominador y el usuario ingrese un "0" el programa no se caiga y le muestre al usuario en donde fue el fallo (por que la división de un numero entre 0 no existe) y así es como se valida la información y se evade el error matemático que cometió el usuario al ingresar ese dato:

```
#include <iostream>
#include <cstdlib>
#include <exception>
using namespace std;

class div_cero : public exception
{
public:
   const char* what() const throw()
   {
      return "Error: división por cero...";
   }
};

int main(int argc, char* argv[])
{
   double N, D;
   cout << "Probando división" << endl;
   cout << "Ingrese el numerador :";
   cin >> N;
   cin.clear();
   cout << "Ingrese el denominador :";
   cin >> D;
   cin.clear();
   try {
      if (D == 0) throw div_cero();
      cout << N << " / " << D << " = " << N / D << endl;
   }
   catch (exception& e) {
      cout << e.what() << endl;
   }
```

```
    system("PAUSE");
    return 0;

}
```

❖ ACTIVIDADES

- Consideraciones: Puede utilizar como IDE: Zinjal, Visual Studio 2019, entre otros. Ello debe ser especificado en su informe.

Nota: Para los siguientes ejercicios, programar de forma ordenada de tal manera que las clases y/o estructuras queden definidas en archivos cabecera (.h), los métodos u operaciones en un archivo .cpp y finalmente el programa principal en otro archivo .cpp. Asimismo, debe utilizar la experiencia de la práctica anterior y debe estar visible en el código.

- **EJERCICIOS PROPUESTOS:**

1. Crear un programa donde exista una función que realice la división de dos números que se le pasan como parámetros y devuelva el resultado. En el caso de que el divisor sea cero se tendrá que generar una excepción que será capturada en la función main.

2. Modificar el ejercicio anterior de la división por 0 para que ahora la excepción generada dentro de la función que realiza la división se trate ahí y luego la relance para que sea tratada en el main también.

3. Para los siguientes casos, proponer y explicar dos ejemplos en C++ considerando:

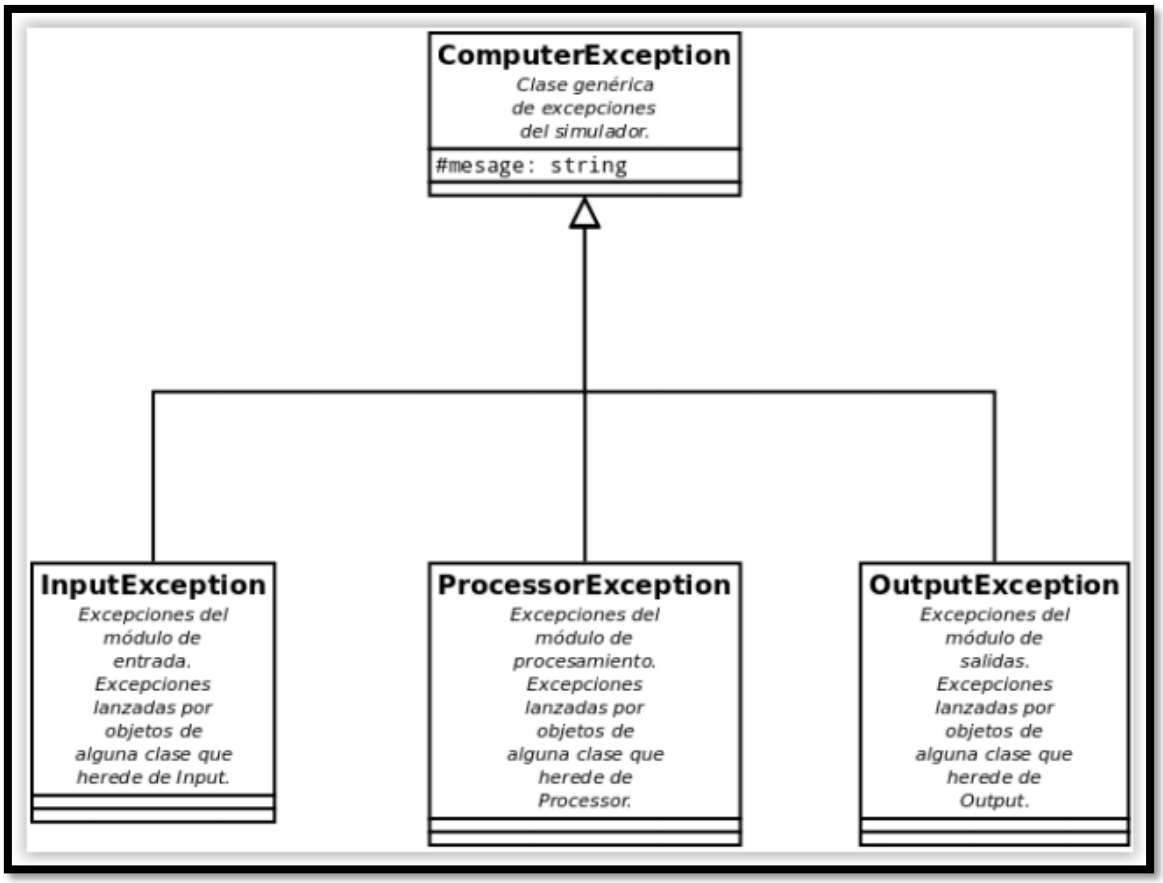
   a) Excepciones provocadas por asignación de memoria insuficiente (generadas por el operador new).
   b) Excepciones por tipos de datos incorrectos (ejemplo, al solicitar un dato numérico el usuario digita letras).

4. Considerar una función raíces que calcula las raíces cuadradas de una ecuación cuadrática. Diseñar una función de modo que se lancen excepciones si no existen raíces reales o el primer coeficiente es cero. El tipo de excepción es error y los valores serán No_raices_reales y primer_coeficiente_cero.

5. Crear una jerarquía de clases de excepciones que permitan controlar los posibles errores que surjan de una manera controlada, clara y elegante. Para ello, crear una clase padre/base ComputerException que tendrá solamente un campo de tipo string mesage y que todas las demás clases de excepciones heredarán de ella. Ahora, crear las clases InputException, ProcessorException y OutputException que manejarán los respectivos errores a esos dispositivos. Por ejemplo, si se tiene un tipo Keyboard que es un fichero del que se van a

leer las entradas, KeyboardException manejará los posibles problemas que pudieran aparecer a la hora de abrir el fichero, leerlo, etc.

Los errores que surjan deberán ser capturados y el programa deberá ser capaz de recuperarse del error siempre que sea posible (si no se puede abrir un fichero para leerlo porque no existe, habrá que volver a pedir la ruta del fichero). En el caso de que no se pueda recuperar de un error, el programa terminará de forma ordenada e indicando con un mensaje claro en qué parte del computador se ha dado el error. Un posible diseño general podría ser el siguiente:



❖ BIBLIOGRAFIA

- Deitel, Paul J., Deitel, Harvey M., "Cómo Programar en C++", 6ta Edición, Ed. Pearson Educación, México 2009.
- Stroustrup, Bjarne, "El Lenguaje de Programación C++", 3ra Edición, Adisson Pearson Educación S.A., Madrid 2002.
- Eckel, Bruce, "Thinking in C++", 2da Edición, Prentice Hall, 2000.