

Clase 10

Excepciones

¿Qué es una excepción?

Es una condición inesperada o inusual que surge durante la ejecución del programa y no puede ser manejada en el contexto local.

Condición anómala e inesperada que necesita ser controlada.

Para que un lenguaje trate excepciones debe proveer mínimamente:

- Un modo de definir las
- Una forma de reconocerlas
- Una forma de lanzarlas y capturarlas
- Una forma de manejarlas especificando el código y respuestas
- Un criterio de continuación

Controlador de excepciones/Manejador: es una sección de código que se encarga de manejar una excepción en un programa. El objetivo es otorgar una forma de recuperarse del error para que no se detenga abruptamente y se siga ejecutando el programa.

Tipos de excepciones

- Implícitas: Definidas por el lenguaje (built-in)
- Explícitas: Definidas por el programador

Los lenguajes deben proveer instrucciones para:

- Definición de la excepción
- Levantamiento de una excepción
- Manejador de la excepción

Después de una excepción se puede:

Continuar la ejecución normal del programa:

- Si después de manejar una excepción el programa puede continuar la ejecución del código restante sin problemas
- El punto de retorno será definido por el lenguaje (por ejemplo, el siguiente bloque de código después del bloque de manejo de excepciones o siguiente instrucción)

Retornar a un estado anterior:

- Cuando el manejo de excepciones puede requerir que el programa regrese a un estado anterior o deshaga acciones realizadas antes de que se produjera la excepción.

Propagar la excepción:

- En el controlador de excepciones no puede manejar completamente la excepción, puede optar por propagarla a un nivel superior en la jerarquía de llamadas.
- El punto de retorno sería el controlador de excepciones en el nivel superior que pueda manejar la excepción o decidir cómo manejarla.

Terminar la ejecución del programa:

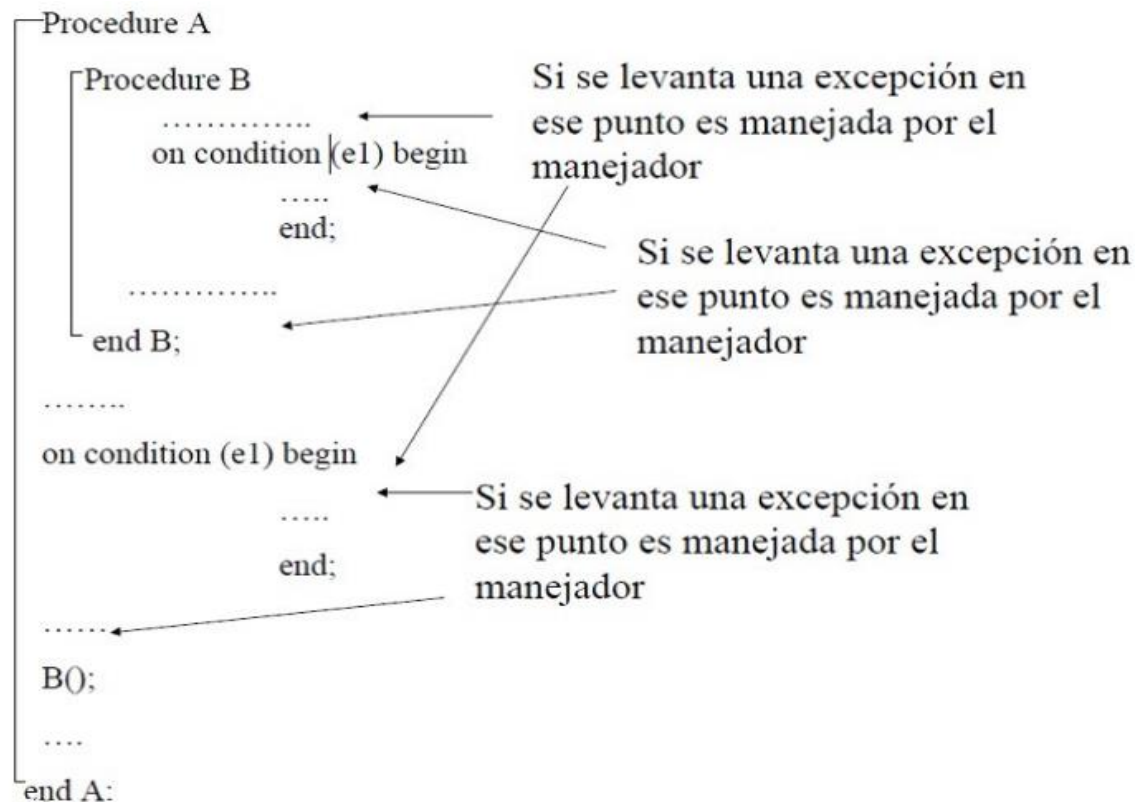
- En situaciones excepcionales o críticas, es posible que el controlador de excepciones determine que no se puede continuar ejecutando el programa de manera segura. El punto de retorno puede ser la finalización del programa o alguna acción específica de cierre antes de la finalización.

Existen dos modelos de ejecución

- **Reasunción:** cuando se produce una excepción, se maneja y al terminar de ser manejada se devuelve el control a la sentencia **siguiente** de donde se levantó la excepción.
 - **PL/1.**
- **Terminación:** cuando se produce una excepción, el bloque donde se levantó la excepción es **terminado** y se ejecuta el manejador asociado a la excepción.
 - **ADA**
 - **CLU**
 - **C++**
 - **Java**
 - **Python**
 - **PHP**

Primer lenguaje que incorpora excepciones – PL/1

- Utiliza el modelo de ejecución (criterio) de **reasunción**.
- Las excepciones se llaman **CONDITIONS**.
- Los manejadores de excepciones se declaran con la sentencia ON:
 - **ON CONDITION(Nombre_excepción) Manejador**
 - El manejador puede ser una instrucción o un bloque de instrucciones.
- Las excepciones son alcanzadas explícitamente con la palabra clave: **Signal condition(Nombre_excepcion)**
- PL/1 tiene excepciones predefinidas (las cuales tienen su manejador asociado). A dichas excepciones las llamamos Built-in exceptions.
 - Ejemplo: zerodivide que se levanta cuando hay una división por cero.
- Los manejadores son ligados de forma dinámica con las excepciones. Una excepción siempre estará ligada con el último manejador definido.
- El alcance de un manejador termina cuando finaliza la ejecución de la unidad donde fue declarado.



Excepciones en ADA

- Usa criterio de terminación.
- Cada vez que se produce una excepción, se termina el bloque dónde se levantó y se ejecuta el manejador asociado, y continúa luego.
- Las excepciones se definen en la zona donde definimos variables y poseen el mismo alcance → `MiExcepcion: exception;`
- Una excepción es alcanzada explícitamente con la palabra clave **raise**.
- Los manejadores se pueden encontrar en el final de cuatro diferentes unidades de programa: Bloque, procedimiento, paquete o tarea.
- La lista de controladores de excepciones lleva el prefijo de la palabra clave `exception`.
- Cada controlador lleva el prefijo de la palabra clave `when`
- `When others` es para manejar una excepción que no definimos. Puede tener efecto colaterales

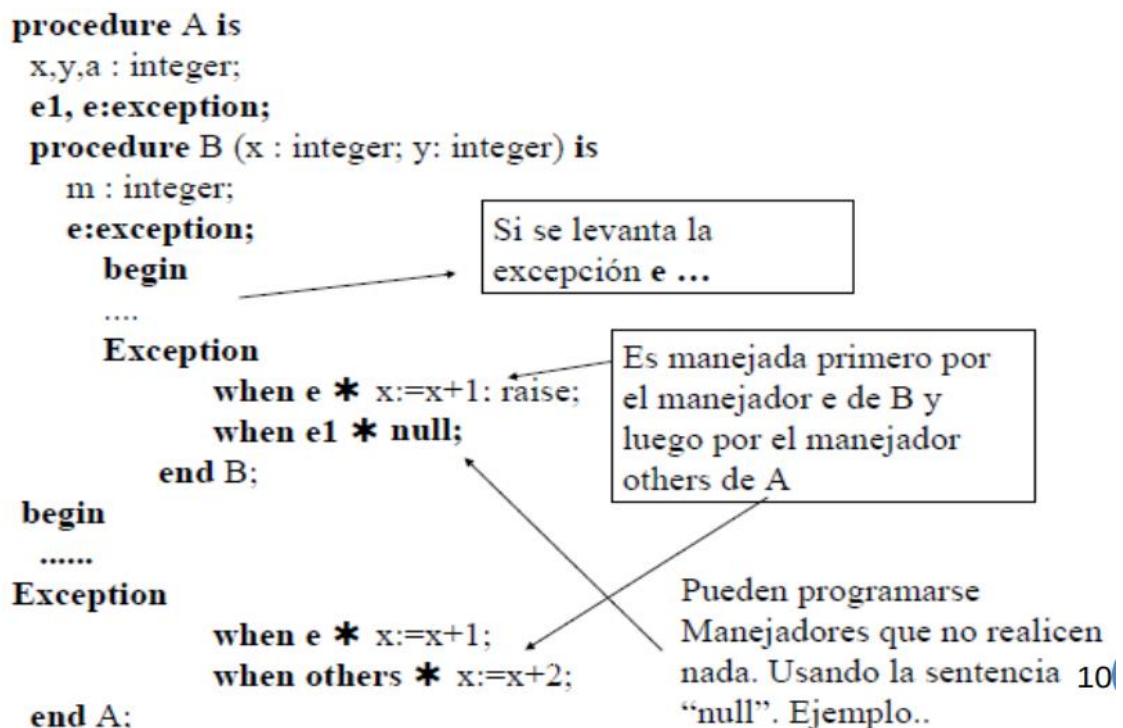
```

Procedure prueba;
  e1, e3: Exception
  Begin
    .....
    Exception
    when e1 → Manejador1;
    when others → ManejadorN;
  End;

```

- Posee excepciones predefinidas:
 - `Constraint_error`, `Program_error`, `Storage_error`, `Numeric_error`, `Name_error`, `Tasking_error`.

- Propagación cuando se produce una excepción:
 - Se termina la unidad, bloque, paquete o tarea donde se levanta la excepción.
 - Si el manejador se encuentra en ese ámbito, se ejecuta.
 - Si el manejador NO se encuentra en dicho ámbito, la excepción es propagada dinámicamente. Se levanta en otro ámbito.
 - Esto provoca que se termine la unidad del otro ámbito.
 - Debemos tomar en cuenta el alcance, la excepción puede volverse anónima.
 - Si el manejador de la excepción no está en el ambito donde se declara la excepción:
 - Se propaga al ambito que llamó mi unidad,bloque,paquete o tarea.
 - Si dicho ámbito tiene para manejar, se maneja la excepción de forma **anónima**, es decir, usando la opción **when others**.
 - Ejemplo: en procedure prueba tenemos e3, supongamos que e3 no tiene manejador y se va a buscar al ámbito que llamó a mi procedure y dicho ámbito tiene manejadores.
 - En dicho ámbito, mi e3 va a entrar al caso donde sea **when others**, por eso decimos que e3 se vuelve anónima.
- Una excepción se puede levantar NUEVAMENTE usando sólo la palabra **raise**. (se propaga y queda como anónima)



- Cuando hacemos **raise** dentro del **procedure B**, se termina el manejador en **B** y trata la excepción **e** (la **e** local al procedimiento **B**) que fue levantada de nuevo en el **others** del manejador en **procedure A** (sale anónima).
 - Si no hay nadie que maneje la excepción, se termina el programa.
- 🚦 Ejemplo en código bien escrito:

```

1  with Ada.Text_IO; use Ada.Text_IO;
2  procedure Main is
3      e : Exception;
4      dividendo : integer;
5      divisor : integer;
6  begin
7      dividendo := 15;
8      divisor := 0;
9      if divisor = 0 then
10         raise e;
11     end if;
12
13     Exception
14         when e => Put("Division por cero");
15         when others => Put("Otro error");
16
17 end Main;

```

- Observar que la excepción posible se define en la parte de variables.
- Como se llega de forma explícita a la excepción, se usa if.

✚ Tomamos en cuenta

- La asociación entre excepción y manejador se da por nombre.
- Si se quiere continuar ejecutando las instrucciones de un bloque que lanzó una excepción, es necesario crear un bloque interno que contenga las instrucciones que pueden fallar junto al manejador de la excepción.

Procedure Bloque () is

....
begin

...
Declare

.....
begin -- del bloque interno
instrucciones que pueden
fallar

exception
manejadores

end; -- del bloque interno

....
Instrucciones que es preciso
ejecutar, aunque se haya
levantado la excepción en el
bloque interno

....
end;

Se pueden
definir
nuevas
excepciones

12

Excepciones en CLU

- Utiliza el criterio de terminación.
- Las excepciones SOLAMENTE pueden ser alcanzadas por procedimientos.
- Las excepciones están asociadas a sentencias.

- Si se produce una excepción en un procedimiento, se va a buscar el manejador a la sentencia asociada.

Procedure ejemplo () signals e1, e2

Begin

.....

if (..) then A(); except

when e1: Manejador1;

when e2 : Manejador2;

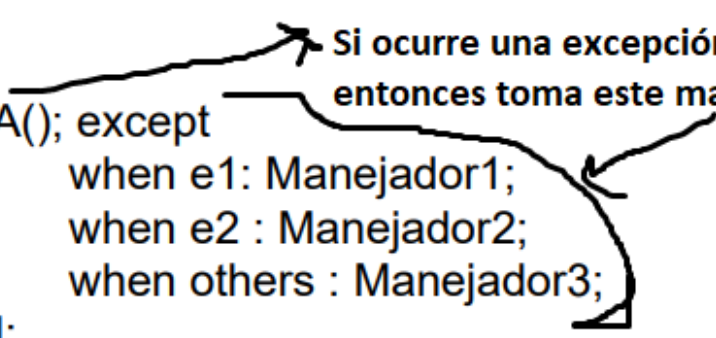
when others : Manejador3;

end;

....

End;

Si ocurre una excepción en A(),
entonces toma este manejador



- Las excepciones que un procedimiento puede lanzar se declaran en el encabezado de dicho procedimiento.
- Son alcanzadas explícitamente con la palabra clave **signal**.
- Los manejadores se colocan al lado de una sentencia simple o compleja.

<sentencia> except

when Nombre-Excepción: Manejador1;

when Nombre-Excepción : Manejador2;

.....

when others: Manejadorn;

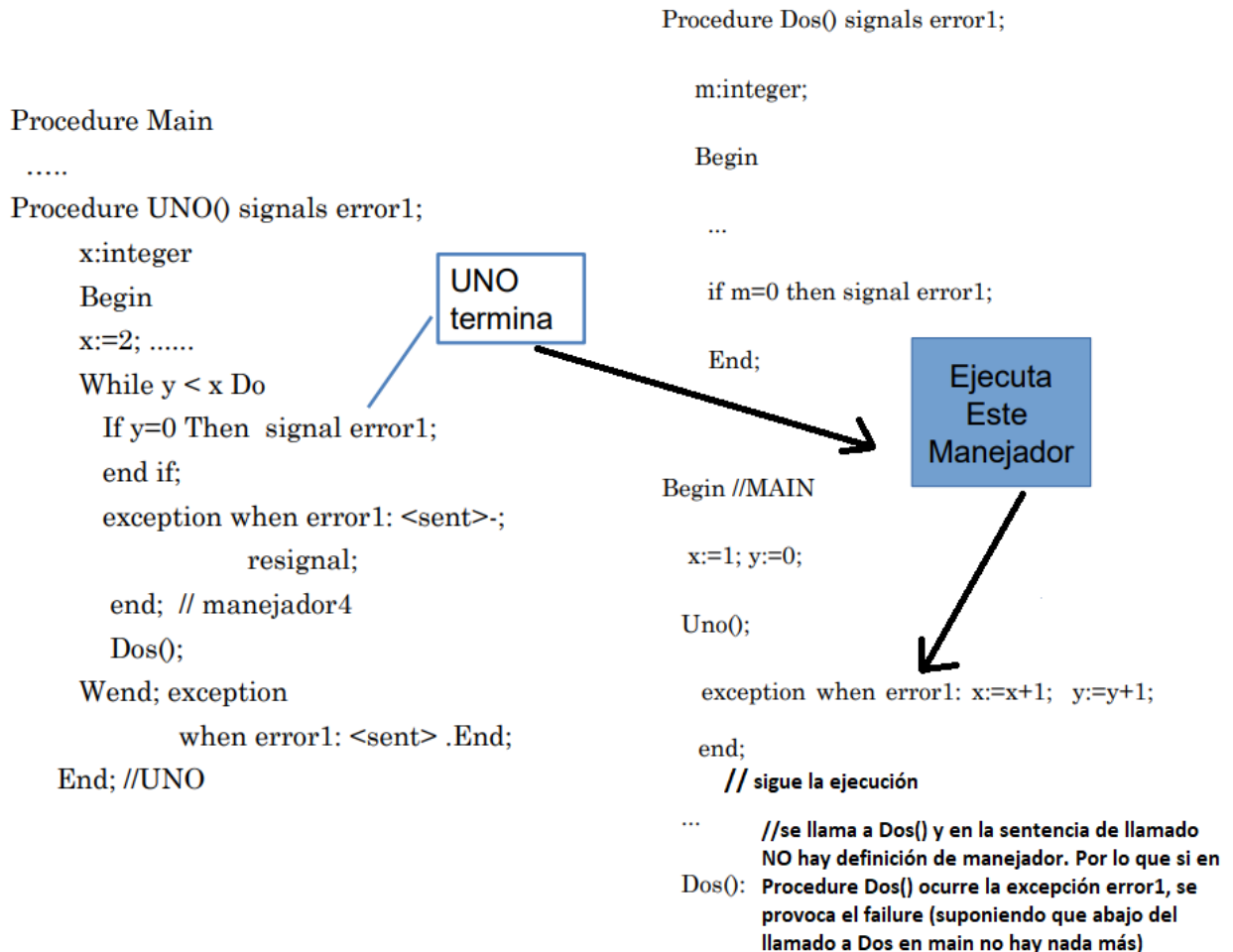
end;

- CLU posee excepciones predefinidas.
- Se pueden pasar parámetros a los manejadores.
- Una excepción se puede levantar de nuevo UNA SOLA VEZ (no como ADA) utilizando **resignal**.
- La excepción se puede levantar en cualquier lugar del código.

🚩 Propagación cuando ocurre una excepción

- El procedimiento donde se levantó la excepción es **TERMINADO**, se devuelve el control al que llamó al procedimiento, justo en la sentencia donde se hace el llamado donde se debe encontrar al manejador.
 - Por esto asociamos manejadores con sentencias.
- Si el manejador está ahí, se ejecuta y luego se pasa el control a la sentencia siguiente a la que está ligada el manejador.
- Si el manejador no se encuentra en ese lugar, se propaga estáticamente la excepción. Es decir, se busca hacia abajo si hay algún manejador que tome la excepción.

- Ocurre una sola propagación dinámica: cuando terminamos el procedimiento al principio.
- Si no hallamos ningún manejador en el procedimiento que hizo el llamado al procedimiento que lanzó la excepción, se levanta la excepción **failure** y termina todo el programa.



- ✚ Otro caso posible:
 - El if dentro de Procedure Uno no es ejecutado, se ejecutan las sentencias hasta que se llama a Dos() dentro de Procedure Uno.
 - La sentencia while tiene asociado un manejador.
 - Si ocurre una excepción en el procedure Dos, va a buscar ese manejador.

Excepciones en C++

- Criterio de terminación.
- Las excepciones pueden alcanzarse explícitamente a través de la sentencia `Throw(nombre_excepcion)`
 - Se pueden pasar parámetros al levantar la excepción.
 - `Throw(Ayuda msg);` msg es el parámetro.
- Tiene excepciones predefinidas.
- Los manejadores de excepciones se asocian a bloques.
 - Conjunto de instrucciones.

- Los bloques que pueden llegar a levantar excepciones van precedidos por la palabra clave **Try**.
- Cuando se finaliza el bloque Try se detallan los manejadores utilizando la palabra clave **Catch(NombreExcepcion)**.
- Cuando se levanta una excepción (Throw dentro del Try) el control del programa se transfiere al manejador correspondiente.

Try

```
{
..... /* Sentencias que pueden provocar una excepción*/
}
catch(NombreExcepción1)
{
..... /* Sentencias Manejador 1*/
}
....
catch(NombreExcepciónN)
{
..... /* Sentencias Manejador N*/
}
...

```

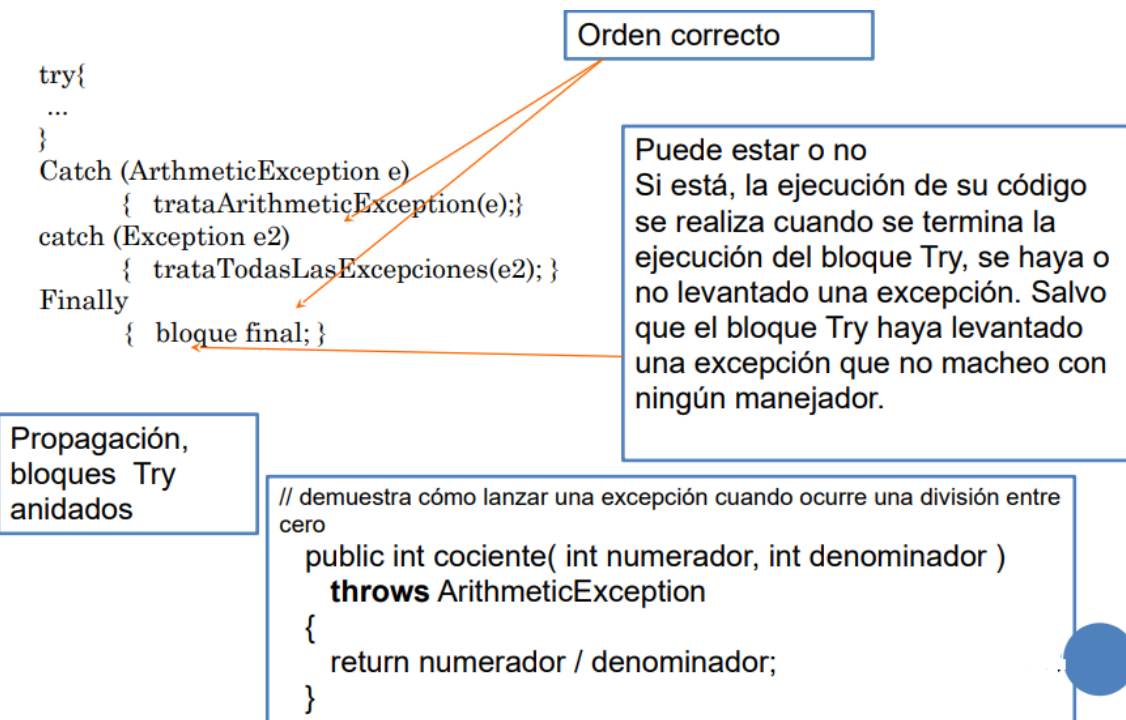
Las excepciones se pueden levantar nuevamente colocando **Throw**.

- Las rutinas pueden listar las excepciones que ellas pueden alcanzar.
 - Void rutina() throw (Ayuda,ZeroDivide);
- ¿Qué sucede si la rutina...?
 - Alcanzó otra excepción no contemplada en el listado.
 - No se propaga la excepción y la función especial unexpected() es ejecutada automáticamente, dicha función normalmente causa abort() que provoca el final del programa.
 - Unexpected puede ser redefinida por el programador.
 - Colocó en su interface el listado de posibles excepciones alcanzables
 - Se puede propagar la excepción. Pero si una excepción se propaga repetidamente y no encuentra manejador, entonces la función terminate() se ejecuta de forma automática.
 - Colocó en su interface una lista vacía -> throw()
 - Ninguna excepción será propagada.

Excepciones en Java

- Igual que C++, las excepciones son objetos alcanzables y manejables por manejadores adicionados al bloque donde se produce la excepción.
- Cada excepción se representa por una instancia de la clase Throwable o de una de sus hijas, Error y Exception.
- La gestión de excepciones usa cinco palabras clave:

- Try
- Catch
- Throw
- Throws: especificación de qué excepción puede enviarse desde un método (posibles excepciones que puede lanzar)
- Finally: bloque de código que se ejecuta siempre
- ✚ Fases del tratamiento:
 - Detectar e informar el error (dos formas):
 - Se lanza la excepción usando throw.
 - Un método detecta una condición anormal que le impide continuar la ejecución y lanza un objeto Excepción.
 - Se recoge el error y se trata (dos formas):
 - Captura de Excepciones -> usando try-catch.
 - Un método recibe un objeto Excepción que le dice que otro método no terminó correctamente su ejecución, entonces se decide como proseguir en función del tipo de error.



Excepciones en Python

- Manejadas con Try except.


```
>>> while True:
...     try:
...         x = int(input("Por favor ingrese un número: "))
...         break
...     except ValueError:
...         print("Oops! No era válido. Intente nuevamente...")
```
- El Try funciona así:
 - Se ejecuta el bloque try.

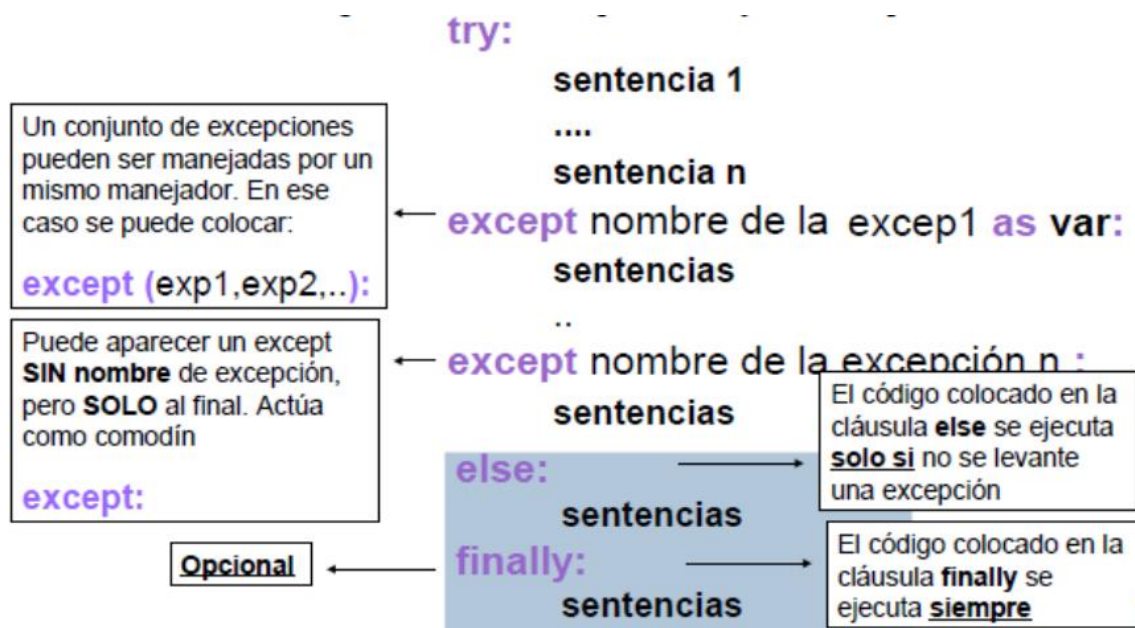
- Si no ocurre ninguna excepción, el bloque except es saltado y se termina la ejecución del bloque try.
- Si ocurre una excepción en el bloque try, el resto del bloque try es saltado. Luego, si el tipo de excepción coincide con alguna excepción nombrada luego de la palabra reservada except, se ejecuta el bloque except que corresponda, y la ejecución del programa continúa.
- Si ocurre una excepción que no coincide con ninguna excepción nombrada en el except, ésta se pasa a declaraciones try de más afuera, si no se encuentra un manejador, significará que es una excepción no manejada, por lo que la ejecución se frena con un mensaje de error.

```

1 dividendo = 15
2 divisor = 0
3 try:
4     resul = dividendo / divisor
5 except(ValueError):
6     print("OOPS")
7
Traceback (most recent call last):
  File "<string>", line 4, in <module>
ZeroDivisionError: division by zero
> |

```

✚ Estructura de manejo de excepciones



✚ ¿Y si una excepción no encuentra un manejador en su bloque try except?

- Búsqueda estática:
 - Analiza si ese try está contenido dentro de otro y si este otro tiene un manejador para la excepción.
- Búsqueda dinámica:
 - Analiza quién llamó y busca allí.

- Si no se halla manejador -> corta el proceso largando el mensaje estándar de error.
- Las excepciones se pueden levantar explícitamente con “raise”

Excepciones en PHP

- Modelo de terminación
- Una excepción puede ser lanzada (thrown) y atrapada (caught).
- El código está dentro de un bloque try.
- Cada bloque try debe tener **al menos** un bloque catch correspondiente.
- Las excepciones pueden ser lanzadas o relanzadas dentro de un bloque catch.
- Se puede utilizar un bloque finally después de los bloques catch.

Como proceder:

- El objeto lanzado DEBE ser una instancia de la clase Exception o de una subclase de la misma. Intentar lanzar un objeto que no corresponda es un error fatal para PHP.
- Cuando una excepción se lanza, el código siguiente a la declaración no se ejecuta. PHP tratará de hallar el primer bloque catch que coincida. Si una excepción no es capturada entonces se emite un error fatal de PHP.
 - Con el mensaje “Excepción no capturada”
 - A menos que se haya definido un gestor con set_exception_handler()

```
<?php
function inverse($x) {
    if (!$x) {
        throw new Exception('División por cero.');
```

El resultado del ejemplo sería:

```
0.2
Primer finally.
Excepción capturada: División por cero.
Segundo finally.
Hola Mundo
```

```
}
return 1/$x;
}
try {
    echo inverse(5) . "\n";
} catch (Exception $e) {
    echo 'Excepción capturada: ', $e->getMessage(), "\n";
} finally {
    echo "Primer finally.\n";
}
try {
    echo inverse(0) . "\n";
} catch (Exception $e) {
    echo 'Excepción capturada: ', $e->getMessage(), "\n";
} finally {
    echo "Segundo finally.\n";
}
// Continuar ejecución
echo 'Hola Mundo\n';
?>
```

Si NO tengo excepciones puedo simularlas mediante uso de If's y llamados a procedimientos que manejen el error.