Lenguaje de Programación I

Msc. Lizeth Joseline Fuentes Pérez

Agenda

1. Errores
2. Excepciones
3. Errores logicos
4. Random
5. Try this

Contenido

Errores de sintaxis

```
int area(int length, int width);

int main()
{
    int s1 = area(7;      // error: falta )
    int s1 = area(7)     // error: falta ;
    Int s3 = area(7);    // error: Int no es un tipo
    int s4 = area('7);  //error: caracter (falta ')
    return area(4,4);
}

int area(int length, int width)
{
    return length*width;
}
```

Errores de tipos

```
int area(int length, int width);

int main()
{
    int x0 = arena(7); // error: funcion no declarada
    int x1 = area(7);  // error: numero de argumentos
    int x2 = area("seven",2); // error: error en el tipo
    // del primer argumento
    return area(4,4);
}

int area(int length, int width)
{
    return length*width;
}
```

No son errores pero...

```
int area(int length, int width);
int main()
{
    int x4 = area(10,-7);  // ok: pero rectangulo
    // con ancho -7?
    int x5 = area(10.7,9.3);// ok: pero llama a
    // area(10,9)
    char x6 = area(100, 9999);// ok, pero trunca
    // el resultado
    return area(4,4);
}
int area(int length, int width)
{
    return length*width;
}
```

Errores en tiempo de Enlance (Link-time).

```
int area(int length, int width);
int main()
{
    int x = area(2,3);
}

//int area(int x, int y) { return x*y; } // es esta

// Funciones con mismo nombre pero diferentes tipos
// no coinciden y son ignoradas.

double area(double x, double y) { return x*y; }

int area(int x, int y, char unit) { return x*y; }
```

Contenido

El que llama a la funcion lidia con errores

```
int area(int length, int width)
{
    return length*width;
}

const int frame_width = 2;
// Calcula el area dentro de un marco
int framed_area(int x, int y)
{
    return area(x-frame_width,y-frame_width);
}
```

El que llama a la funcion lidia con errores

```
int main()
try
{
    int x = 1;
    int y = 2;

    {
        if (x<=0) throw runtime_error("x es negativo");
        if (y<=0) throw runtime_error("y es negativo");
        int area1 = area(x,y);
    }
```

El que llama a la funcion lidia con errores

```
{// Si no separamos especificamente el error seria:
    if (x<=0 || y<=0) throw
        runtime_error("argumento no positivo");
        // || significa "o"
    int area1 = area(x,y);
    if (z<=2)
        throw
            runtime_error("argumento no positivo");
    int area2 = framed_area(1,z);

    if (y<=2 || z<=2)
        throw
            runtime_error("argumento no positivo");
    int area3 = framed_area(y,z);
    double ratio = double(area1)/area3;
    //convertir a double
}
```

El que llama a la funcion lidia con errores

```
        {
            if (1-frame_width<=0 || z-frame_width<=0)
                throw runtime_error("argumento negativo");
            int area2 = framed_area(1,z);
            if (y-frame_width<=0 || z-frame_width<=0)
                throw runtime_error("argumento negativo");
            int area3 = framed_area(y,z);
        }
    }
    catch (exception & e) {
        cerr << "error: " << e.what() << '\n';
        return 1;
    }
```

Los errores se manejan dentro de la funcion

```
int area(int length, int width);
int framed_area(int x, int y)
{
    const int frame_width = 2;
    if (x-frame_width<=0 || y-frame_width<=0)
        throw runtime_error("argumento negativo");
    return area(x-frame_width,y-frame_width);
}

int area(int length, int width)
{
    if (length<=0 || width <=0)
        throw runtime_error("argumento negativo");
    return length*width;
}
```

Los errores se manejan dentro de la funcion

```
int main()
try
{
    int x = -1;
    int y = 2;
    int z = 4;

    int area1 = area(x,y);
    int area2 = framed_area(1,z);
    int area3 = framed_area(y,z);
    double ratio = double(area1)/area3;
    // convertir a double
}
catch (exception& e) {
    cerr << "error: " << e.what() << '\n';
    return 1;
}
```

Usando nuevos tipos

```
class Bad_area { }; // Nuevo tipo de dato
int area(int length, int width)
{
if (length<=0 || width<=0) throw Bad_area{};
return length*width;
}
int main()
try
{
    int x = -1;
    int y = 2;
    int z = 4;
    int area1 = area(x,y);
    int area2 = framed_area(1,z);
}
catch (Bad_area) {
cout << "Oops! argumentos invalidos para area()\n";
}
```

Errores de rango

```
int main()
try {
    vector<int> v;
    for (int x; cin>>x; )
        v.push_back(x);

    for (int i = 0; i<=v.size(); ++i)
        cout << "v[" << i <<"] == " << v[i] << '\n';
    } catch (out_of_range) {
    cerr << "Oops! Error de rango\n";
    return 1;
}
catch (...)
{
    cerr << "Error: algo salio mal\n";
    return 2;
}
```

Contenido

Errores de logica

```
int main()
{
    double temp = 0;
    double sum = 0;
    double high_temp = 0;   // inicializa
    double low_temp  = 0;   // inicializa
    int no_of_temps  =  0;

    while (cin>>temp) {          // lee temp
        ++no_of_temps;           // cuenta temps
        sum += temp;             // calcula la suma
        // Encuentra la mayor
        if (temp > high_temp) high_temp = temp;
        // Encuentra la menor
        if (temp < low_temp)  low_temp  = temp;
    }
//-----------------------------
```

Errores de logica

```
    //.......
        cout << "Temperatura alta: "     << high_temp
<< endl;
        cout << "Temperatura baja: "      << low_temp
<< endl;
        cout << "Temperatura promedio: "
        << sum/no_of_temps << endl;
        return 0;
}
```

Contenido

Generando numeros aleatorios

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main ()
{
   int num_random;
   srand (time(NULL));//incializa la semilla
   num_random = rand() % 10 + 1;
   printf("%d",num_random);
   return 0;
}
```

Contenido

Try this

Implementa los siguientes codigos :

1. Utilizando el codigo de generacion de numeros aleatorios, implementa el siguiente juego.
   Vacas y toros : El programa tiene un vector que contiene 4 digitos diferentes (Ej : 3567 pero no 3377). El usuario debe descubrir los numeros, adivinando varias veces. Digamos que el numero a adivinar es 1234, y el usuario adivina 1359 ; la respuesta seria **1 toro y 1 vaca**, porque adivino correctamente 1 digito (1),en la posicion correcta (eso es un toro), y adivino correctamente el digito (3) pero en la posicion equivocada (eso es una vaca). El programa continua hasta que el usuario obtiene 4 toros.
   Haz que el usuario una vez que adivina pueda jugar de nuevo, entonces el numero a adivinar debe cambiar.