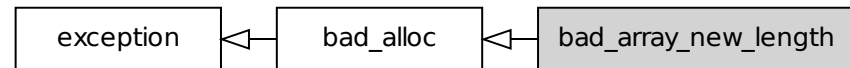

C++ EXCEPCIONES

Excepciones

- Permiten representar errores en tiempo de ejecución.
- Pueden producirse al usar `dynamic_cast`, `typeid`, `new`, `std::vector::at`, `std::string::substr`, etc.
- Las excepciones generadas por la biblioteca estándar heredan de **`std::exception`**. (<https://es.cppreference.com/w/cpp/language/exceptions>)
- Ejemplos:
 - **`out_of_range`** : Se intenta acceder a elementos fuera del rango definido.
 - **`invalid_argument`**: Cuando un valor de argumento no ha sido aceptado.
 - **`bad_alloc`**: Lanzado por las funciones de asignación de memoria para reportar una falla al asignar almacenamiento.

Excepciones

```
1 int main() {  
2     int *x = NULL;  
3     int y = 1000000000;  
4     x = new int[y];  
5     delete[] x;  
6     return 0;  
7 }
```



```
terminate called after throwing an instance of 'std::bad_array_new_length'  
what():  std::bad_array_new_length
```

- Se trata de una referencia a un objeto *bad_alloc*, que es el asociado a excepciones consecuencia de aplicar el operador *new*.

Captura de excepciones

- Cuando se producía una excepción, hasta ahora se *terminaba el programa*.
- Los errores o "situaciones de excepción" son problemas comunes con los que deben lidiar los programas, y no es una buena medida que el programa termine abruptamente cuando se produce un error.
- Se necesita un mecanismo para *capturar excepciones* y recuperarse del error, o bien, mostrar un mensaje amigable al usuario.

Try...catch

- La captura de excepciones se logra con la estructura *try ... catch*:

```
1 try
2 {
3     // Código normal
4 }
5 catch(const std::exception& ex)
6 {
7     // Código para manejar el caso de excepción
8 }
```

Try...catch

- Catch-clause que declara un parámetro formal con nombre:

```
try { /* */ } catch (const std::exception& e) { /* */ }
```

- Catch-clause que declara parámetro sin nombre:

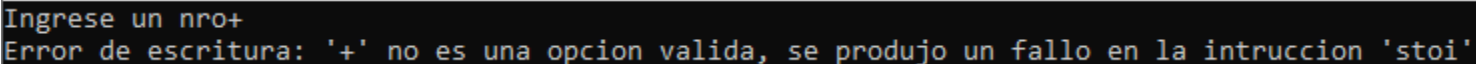
```
try { /* */ } catch (const std::exception&) { /* */ }
```

- Catch-all handler, que es activo para cualquier excepción:

```
try { /* */ } catch (...) { /* */ }
```

Ejemplo: Try/catch

```
1 string opc;
2 cout<<"Ingrese un nro";
3 cin >> opc;
4 try
5 {
6     stoi(opc);
7 }
8 catch (const invalid_argument &ex)
9 {
10     cout << "Error de escritura: '" << opc << "' no es una opcion
11     valida, se produjo un fallo en la instruccion '"<< ex.what()<<"'\n";
12}
```



The screenshot shows a terminal window with a black background and green text. It displays the program's output for the input '+'. The first line is 'Ingrese un nro+' and the second line is 'Error de escritura: '+' no es una opcion valida, se produjo un fallo en la instruccion 'stoi''.

Ejemplo: Try/catch

```
1  vector<int> primos= {2,3,5,7,11,13,17,19,23,29};
2  try
3  {   int i;
4
5      cout<<"Ingrese el numero de primo que quiere consultar (1-10):";
6      cin>>i;
7      int n=primos.at(i-1);
8      cout<< "El numero primo " << i << " es el " << n;
9  }
10 catch(out_of_range &ex)
11 {
12     cout<<"El numero que ingreso no esta entre 1 y 10";
13 }
```

```
Ingrese el numero de primo que quiere consultar (1-10):
13
El numero que ingreso no esta entre 1 y 10

Process returned 0 (0x0)   execution time : 8.160 s
Press any key to continue.
```


Ejemplo: Try/catch

```
1 vector<int> primos= {2,3,5,7,11,13,17,19,23,29};
2 bool listo=false;
3 while(listo==false)
4 {   try {
5
6       int i;
7
8       cout<<"Ingrese el numero de primo que quiere consultar (1-10):\n";
9       cin>>i;
10      int n=primos.at(i-1);
11      cout<< "El numero primo " << i << " es el " << n;
12      listo=true; }
13  catch(out_of_range &ex) {
14      cout<<"El numero que ingreso no esta entre 1 y 10\n"; }
15 }
```

```
Ingrese el numero de primo que quiere consultar (1-10):
13
El numero que ingreso no esta entre 1 y 10
Ingrese el numero de primo que quiere consultar (1-10):
```

Ejemplo: Varios tipos de error

```
1 vector<int> primos= {2,3,5,7,11,13,17,19,23,29};
2 bool listo=false;
3 while(listo==false)
4 {
5     try {
6         int i;
7         cout<<"Ingrese el numero de primo que quiere consultar (1-10):\n";
8         cin>>i;
9         int n=primos.at(i-1);
10        cout<< "El numero primo " << i << " es el " << n;
11        listo=true; }
12    catch(...) {
13        cout<< "Se produjo un error inesperado\n"; }
14 }
```

```
Ingrese el numero de primo que quiere consultar (1-10):
13
Se produjo un error inesperado
Ingrese el numero de primo que quiere consultar (1-10):
```

Ejemplo: Varios tipos de error

```
1 vector<int> primos= {2,3,5,7,11,13,17,19,23,29};
2 bool listo=false;
3 while(listo==false)
4 { try {
5     int i;
6     cout<<"Ingrese el numero de primo que quiere consultar (1-10):\n";
7     cin>>i;
8     int n=primos.at(i-1);
9     cout<< "El numero primo " << i << " es el " << n;
10    listo=true; }
11 catch(exception &e) {
12    cout<< "Se produjo el error inesperado\n"<<e.what()<<"\n"; }
13 }
14 }
15 }
```

```
Ingrese el numero de primo que quiere consultar (1-10):
13
Se produjo un error inesperado
vector::_M_range_check: __n (which is 12) >= this->size() (which is 10)
Ingrese el numero de primo que quiere consultar (1-10):
```

Lanzado de excepciones

- Las excepciones se lanzan utilizando la instrucción *throw*.
- Una excepción es un objeto de una clase que representa una situación anormal.
- Para lanzar una excepción, hace falta crear un objeto del tipo de excepción que se quiere lanzar:

```
throw TipoException(<parametros del constructor>);
```

- Cuando se lanza una excepción se corta la ejecución del método y se busca si se produjo en el contexto de un try ... catch que capture ese tipo.
- En caso de que no sea capturada la excepción, se termina la aplicación con un mensaje de error.

Ejemplo: Lanzado de excepciones

```
1 int compare( int a, int b )
2 {
3     if ( a < 0 || b < 0 )
4     {
5         throw invalid_argument("Valores negativos recibidos");
6     }
7 }
```

```
terminate called after throwing an instance of 'std::invalid_argument'
  what():  Valores negativos recibidos
```

Ejemplo: Lanzado de excepciones

- Su uso será:

```
1 try
2 {
3     compare( -1, 3 );
4 }
5 catch(const invalid_argument &e)
6 {
7     cout<<"Se produjo un error en los argumentos: "<< e.what();
8 }
```

```
Se produjo un error en los argumentos: Valores negativos recibidos
Process returned 0 (0x0)   execution time : 3.271 s
Press any key to continue.
```

Ejemplo: Lanzado de excepciones

```
1  try
2  {
3      compare( -1, 3 );
4  }
5  catch(const invalid_argument &e)
6  {
7      cout<<"Se produjo un error en los argumentos, volvemos a intentar...\n";
8      compare( -1, 3 );
9  }
10 catch(const invalid_argument &e)
11 {
12     throw;
13 }
```

Se produjo un error en los argumentos, volvemos a intentar...
terminate called after throwing an instance of 'std::invalid_argument'
what(): Valores negativos recibidos

Ejemplo: Lanzado de excepciones

```
try
{
    compare( -1, 3 );
}
catch(exception &e)
{
    cout<<"Se produjo un error \n"<<e.what();
}
}
```

```
Se produjo un error
Valores negativos recibidos
Process returned 0 (0x0)   execution time : 3.180 s
Press any key to continue.
```


Ejemplo: Lanzado de excepciones

```
int main ()
{
    try
    {
        throw 20;
    }
    catch (int e)
    {
        cout << "Ocurrio una excepcion. Nro de Excepcion" << e << '\n';
    }
    return 0;
}
```

Ejemplo: Lanzado de excepciones

```
1 try
2 {
3     // code here
4 }
5 catch (int param) { cout << "int exception"; }
6 catch (char param) { cout << "char exception"; }
7 catch (...) { cout << "default exception"; }
```

Ejemplo: Lanzado de excepciones

```
1  try
2  {
3      try
4      {
5          //code here
6      }
7      catch (int n)
8      {
8          throw;
9      }
10 }
11 catch (...) {
12     cout << "Ocurrio una Excepcion"; }
```

Ejemplo: Creación de excepciones

```
1 class myexception: public exception
2 {
3     virtual const char* what() const throw()
4     {
5         return "Se produjo mi excepcion";
6     }
7 } myex;
```

Ejemplo: Creación de excepciones

```
1 int main () {  
2     try  
3     {  
4         throw myex;  
5     }  
6     catch (exception& e)  
7     {  
8         cout << e.what() << '\n';  
9     }  
10    return 0;  
11 }
```

Se produjo mi excepcion

Process returned 0 (0x0) execution time : 3.344 s
Press any key to continue.

Fin
