**Error handling**

El manejo de excepciones es una técnica de programación que permite al programador controlar los errores ocasionados durante la ejecución de un programa informático. Cuando ocurre cierto tipo de error, el sistema reacciona ejecutando un fragmento de código que resuelve la situación, por ejemplo retornando un mensaje de error o devolviendo un valor por defecto.

Una excepción en términos de lenguaje de programación es la indicación de un problema que ocurre durante la ejecución de un programa. Sin embargo, la palabra excepción se refiere a que este problema ocurre con poca frecuencia generalmente cuando existe algún dato o instrucción que no se apega al funcionamiento del programa por lo que se produce un error. El manejo de excepciones permite al usuario crear aplicaciones tolerantes a fallas y robustas (resistentes a errores) para controlar estas excepciones y que pueda seguir ejecutándose el programa sin verse afectado por el problema. En lenguaje java estas excepciones pueden manejarse con las clases que extienden el paquete Throwable de manera directa o indirecta, pero existen diversos tipos de excepciones y formas para manejarlas.

**Uso del manejo de excepciones**

El manejo de excepciones ayuda al programador a trasladar el código para manejo de errores de la línea principal de ejecución, además se puede elegir entre manejar todas las excepciones, las de cierto tipo o de las de grupos relacionados, esto hace que la probabilidad de pasar por alto los errores se reduzca y a la vez hace los programas más robustos. Pero es importante utilizar un lenguaje de programación que soporte este manejo, de lo contrario el procesamiento de errores no estará incluido y hará el programa más vulnerable. Este manejo está diseñado para procesar errores que ocurren cuando se ejecuta una instrucción, algunos ejemplos son: desbordamiento aritmético, división entre cero, parámetros inválidos de método y asignación fallida en la memoria. Sin embargo, no está diseñado para procesar problemas con eventos independientes al programa como son pulsar una tecla o clic al mouse.

Las excepciones se dividen en verificadas y no verificadas. Es importante esta división porque el compilador implementa requerimientos de atrapar o declarar para las verificadas lo que hará que se detecten las excepciones automáticamente y de acuerdo al lenguaje de programación utilizado se utilizará un método para corregirlas. Sin embargo para las no verificadas se producirá un error indicando que deben atraparse y declararse. Por eso el programador debe pensar en los problemas que pueden ocurrir cuando se llama a un método y definir excepciones para verificarse cuando sean importantes. Las clases de excepciones pueden derivarse de una superclase común, por lo que con un manejador para atrapar objetos de la superclase, también se pueden atrapar todos los objetos de las subclases de esa clase. Pero también, se pueden atrapar a cada uno de los tipos de las subclases de manera individual si estas requieren ser procesadas diferente.

A cada célula se le conoce como compiladora de distintos.

**Limpieza de pila**

En ocasiones cuando se lanza una excepción, pero no se atrapa en un enlace específico, la pila de llamadas se limpia y el programa intenta volverlo a atrapar en el siguiente bloque, esto se conoce como limpieza de pila. Este proceso hace que el método en el que no se atrapó la excepción termine, todas sus variables quedan fuera del enlace y el control regresa a la instrucción que originalmente la invocó. La limpieza de pila se repetirá hasta que la excepción pueda ser atrapada porque de lo contrario se producirá un error a la hora de compilar.

**Aserciones**

Las aserciones ayudan a asegurar la validez del programa al atrapar los errores potenciales e identificar los posibles errores lógicos del desarrollo. Estas pueden escribirse como comentarios para apoyar a la persona que desarrolla el programa. Algunos ejemplos son:

**Precondiciones y pos condiciones**

Estas características son utilizadas por los programadores para hacer un análisis de lo esperado del programa antes y después de su ejecución. Son importantes porque gracias a ellas se pueden detectar posibles fallas en el programa y corregirlas.

Las precondiciones son verdaderas cuando se invoca a un método, estas describen las características del método y las expectativas que se tienen en el estado actual del programa. Si no se cumplen las precondiciones el comportamiento del método es indefinido por lo que se lanza una excepción que esté preparada o continuar con el programa esperando el error. Las pos condiciones describen las restricciones en el entorno y cualquier efecto secundario del método. Es recomendable escribirlas para saber que esperar en un futuro si es que se hacen modificaciones.

**Conclusión**

El manejo de excepciones ayuda a lidiar con los errores de una aplicación por medio de la manipulación del código para hacer programas más robustos. Además existen herramientas que ayudan a manejarlas tal es el caso de los bloques try (intentar) que encierran el código que puede lanzar una excepción y los bloques el catch (atrapar) que lidian con las excepciones que surjan. También existen técnicas que el programador utiliza para conocer el posible funcionamiento del programa y detectar los errores que pueda contener.

**Lenguaje C**

C es un lenguaje de programación originalmente desarrollado por Dennis Ritchie entre 1969 y 1972 en los Laboratorios Bell, como evolución del anterior lenguaje B, a su vez basado en BCPL.

Al igual que B, es un lenguaje orientado a la implementación de Sistemas operativos, concretamente Unix. C es apreciado por la eficiencia del código que produce y es el lenguaje de programación más popular para crear software de sistemas, aunque también se utiliza para crear aplicaciones.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel, ya que dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portable entre plataformas y/o arquitecturas.

**Características**

**Propiedades**

Un núcleo del lenguaje simple, con funcionalidades añadidas importantes, como funciones matemáticas y de manejo de archivos, proporcionadas por bibliotecas.

Es un lenguaje muy flexible que permite programar con múltiples estilos. Uno de los más empleados es el estructurado "no llevado al extremo" (permitiendo ciertas licencias de ruptura).

Un sistema de tipos que impide operaciones sin sentido.

Usa un lenguaje de preprocesado, el preprocesador de C, para tareas como definir macros e incluir múltiples archivos de código fuente.

Acceso a memoria de bajo nivel mediante el uso de punteros.

Interrupciones al procesador con uniones.

Un conjunto reducido de palabras clave.

Por defecto, el paso de parámetros a una función se realiza por valor. El paso por referencia se consigue pasando explícitamente a las funciones las direcciones de memoria de dichos parámetros.

Punteros a funciones y variables estáticas, que permiten una forma rudimentaria de encapsulado y polimorfismo.

Tipos de datos agregados (struct) que permiten que datos relacionados (como un empleado, que tiene un id, un nombre y un salario) se combinen y se manipulen como un todo (en una única variable "empleado").

**Carencias**

Recolección de basura nativa, sin embargo se encuentran a tal efecto bibliotecas como la "libgc" desarrollada por Sun Microsystems, o el Recolector de basura de Boehm.

Soporte para programación orientada a objetos, aunque la implementación original de C++ fue un preprocesador que traducía código fuente de C++ a C. Véase también la librería GObject.

Funciones anidadas, aunque GCC tiene esta característica como extensión.

Soporte nativo para programación multihilo. Disponible usando librerías como libpthread.

Aunque la lista de las características útiles de las que carece C es larga, este factor ha sido importante para su aceptación, porque escribir rápidamente nuevos compiladores para nuevas plataformas, mantiene lo que realmente hace el programa bajo el control directo del programador, y permite implementar la solución más natural para cada plataforma. Ésta es la causa de que a menudo C sea más eficiente que otros lenguajes. Típicamente, sólo la programación cuidadosa en lenguaje ensamblador produce un código más rápido, pues da control total sobre la máquina, aunque los avances en los compiladores de C y la complejidad creciente de los microprocesadores modernos han reducido gradualmente esta diferencia.

En algunos casos, una característica inexistente puede aproximarse. Por ejemplo, la implementación original de C++ consistía en un preprocesador que traducía código fuente C++ a C. La mayoría de las funciones orientadas a objetos incluyen un puntero especial, que normalmente recibe el nombre "this", que se refiere al objeto al que pertenece la función. Mediante el paso de este puntero como un argumento de función, esta funcionalidad puede desempeñarse en C. Por ejemplo, en C++ se puede escribir:

stack.push(val);

Mientras que en C se podría escribir:

push(stack, val);

Donde el argumento stack es un puntero a una struct equivalente al puntero this de C++, que es un puntero

**Proceso de compilación**

La compilación de un programa C se realiza en varias fases que normalmente son automatizadas y ocultadas por los entornos de desarrollo:

Preprocesado consistente en modificar el código fuente en C según una serie de instrucciones (denominadas directivas de preprocesado) simplificando de esta forma el trabajo del compilador. Por ejemplo, una de las acciones más importantes es la modificación de las inclusiones (#include) por las declaraciones reales existentes en el archivo indicado.

Compilación que genera el código objeto a partir del código ya preprocesado.

Enlazado que une los códigos objeto de los distintos módulos y bibliotecas externas (como las bibliotecas del sistema) para generar el programa ejecutable final.o a un objeto.

**Bibliotecas**

Una biblioteca de C es una colección de funciones utilizadas en el lenguaje de programación C. Las bibliotecas más comunes son la biblioteca estándar de C y la biblioteca del estándar ANSI C, la cual provee las especificaciones de los estándares que son ampliamente compartidas entre bibliotecas. La biblioteca ANSI C estándar, incluye funciones para la entrada y salida de archivos, alojamiento de memoria y operaciones con datos comunes: funciones matemáticas, funciones de manejo de cadenas de texto y funciones de hora y fecha.

Otras bibliotecas C son aquellas utilizadas para desarrollar sistemas Unix, las cuales proveen interfaces hacia el núcleo. Estas funciones son detalladas en varios estándares tales como POSIX y el Single UNIX Specification.

Ya que muchos programas han sido escritos en el lenguaje C existe una gran variedad de bibliotecas disponibles. Muchas bibliotecas son escritas en C debido a que C genera código objeto rápido; los programadores luego generan interfaces a la biblioteca para que las rutinas puedan ser utilizadas desde lenguajes de mayor nivel, tales como Java, Perl y Python.

**Assembler**

El lenguaje ensamblador, o assembler (en inglés assembly language y la abreviación asm), es un lenguaje de programación de bajo nivel. Consiste en un conjunto de mnemónicos que representan instrucciones básicas para los computadores, microprocesadores, microcontroladores y otros circuitos integrados programables. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portátiles.

Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina.

Muchos sofisticados ensambladores ofrecen mecanismos adicionales para facilitar el desarrollo del programa, controlar el proceso de ensamblaje, y la ayuda de depuración. Particularmente, la mayoría de los ensambladores modernos incluyen una facilidad de macro (descrita más abajo), y se llaman macro ensambladores.

Fue usado principalmente en los inicios del desarrollo de software, cuando aún no se contaba con potentes lenguajes de alto nivel y los recursos eran limitados. Actualmente se utiliza con frecuencia en ambientes académicos y de investigación, especialmente cuando se requiere la manipulación directa de hardware, alto rendimiento, o un uso de recursos controlado y reducido. También es utilizado en el desarrollo de controladores de dispositivo (en inglés, device drivers) y en el desarrollo de sistemas operativos, debido a la necesidad del acceso directo a las instrucciones de la máquina. Muchos dispositivos programables (como los microcontroladores) aún cuentan con el ensamblador como la única manera de ser manipulados.