Universidad de Jaén

Departamento de Informática

Prácticas de Estructuras de Datos
Grado en Ingeniería en Informática
Curso 2023/2024

Práctica 1. Introducción a los contenedores: vectores

Sesiones de prácticas: 1

Introducción

Entre las entidades de una aplicación es habitual encontrarse con relaciones de multiplicidad superior a 1 que deben gestionarse mediante alguna estructura de datos que permita almacenar una colección de elementos. A estas colecciones se les suele denominar, de forma genérica, Contenedores, y suelen disponer de funcionalidades similares. En esta práctica, que será incremental, implementaremos y usaremos un vector básico de C que, en la siguiente sesión, será mejorado añadiendo funcionalidad extra hasta convertirlo en un vector dinámico. Para ello, se hará una ampliación del enunciado.

Además, veremos como en los lenguajes con tipos, las plantillas, patrones, o *templates* en inglés, son un recurso sintáctico muy cómodo para adaptar un contenedor de forma que permita utilizarse con diferentes tipos de datos.

Objetivos

1. Implementar la clase VDinamico<T> utilizando **patrones de clase y excepciones**. Se encargará de almacenar y recuperar elementos de forma segura del vector. El contenedor utilizará la memoria dinámica para almacenar los datos en el vector y excepciones para notificar situaciones anómalas, como el acceso a una posición no válida.
2. Elaborar un programa de prueba para comprobar su correcto funcionamiento.
3. Instalar el entorno de desarrollo C++ CLion en el equipo personal de trabajo.

Descripción de la EEDD

Implementar la clase VDinamico<T> para que tenga la siguiente funcionalidad utilizando patrones de clase y excepciones.

- Constructor por defecto VDinamico<T>(unsigned int n), que crea espacio para n datos. Constructor copia VDinamico<T> (const VDinamico<T> & origen).
- Constructor de copia parcial VDinamico<T> (const VDinamico<T> &origen, unsigned int posicionInicial, unsigned int numElementos). En este constructor hay que tener en cuenta que el vector que se genera debe tener un tamaño máximo de numElementos.
- Operador de asignación (=).
- Operador [] para acceder a un dato para lectura/escritura.
- Ordenar el vector de menor a mayor. Se puede utilizar la función sort de <algorithm>: void ordenar().
- Ordenar el vector de mayor a menor con la función `void ordenarRev()`.
- El destructor correspondiente.

Programa de prueba: Gestión de un vector dinámico de enteros

El programa de prueba consiste en instanciar el vector dinámico al tipo int para gestionar elementos de forma que podamos añadir dinámicamente valores.

La prueba implementada en la función main() consistirá en:

- Instanciar un vector de enteros de tamaño un millón y almacenar el mismo número de valores enteros aleatorios sin repetidos. Mostrar el contenido de las primeras 20 posiciones del vector.
- Ordenar el vector de mayor a menor y mostrar la información de sus primeras 20 posiciones.
- Ordenar el vector de menor a mayor y mostrar la información de sus primeras 20 posiciones.

Estilo y requerimientos del código:

1. El código debe ser claro, tener un estilo definido y estar perfectamente indentado, para ello se pueden seguir algunos de los estilos preestablecidos para el lenguaje C++ (http://geosoft.no/development/cppstyle.html).

2. Deben comprobarse todos los posibles errores y situaciones de riesgo que puedan ocurrir (desbordamientos de memoria, parámetros con valores no válidos, etc.) y lanzar las excepciones correspondientes, siempre que tenga sentido. Leer el tutorial de excepciones disponible en el repositorio de la asignatura en docencia virtual.

3. Se valorará positivamente la calidad general del código: claridad, estilo, ausencia de redundancias, etc.