**Gestión de Memoria estática con TDA**

1. **Definición.**

Para implementar alguna estructura de datos, primero es necesario tener muy claro cómo va a ser el manejo de memoria.

La diferencia entre estructuras estáticas y dinámicas está en el manejo de memoria.

En la memoria estática durante la ejecución del programa el tamaño de la estructura no cambia.

La estructura que maneja memoria estática son los vectores. Un vector es una colección finita, homogénea y ordenada de elementos. Un vector es una colección finita, homogénea y ordenada de elementos. Es finita porque todo arreglo tiene un límite, homogénea porque todos los elementos son del mismo tipo y ordenada porque se puede determinar cuál es el enésimo elemento.

Un vector tiene dos partes: Componente e índices.

Los componentes hacen referencia a los elementos que forman el arreglo y los índices permiten referirse a los componentes del arreglo en forma individual.

* Los arreglos se clasifican en:
* Unidimensionales (vectores o listas).
* Bidimensionales (matrices o tablas).
* Multidimensionales

Ejemplo TDA con Memoria Estática:

***main.cpp***

#include <iostream>  
#include "Vector.h"  
#include "Validacion.h"  
**int** **main**(){  
 std::cout << "**\n\t**MEMORIA ESTATICA CON TDA**\n\n**";  
 **int** tamanoVector = ingresar\_enteros("Ingresar el tamanio de la vector: ");  
 Vector vec(tamanoVector);  
 **do** {  
 system("cls");  
 std::cout << "**\n\t**MEMORIA ESTATICA CON TDA**\n\n**" <<  
 "1. Agregar**\n**" <<  
 "2. Eliminar**\n**" <<  
 "3. Mostrar**\n**" <<  
 "4. Salir**\n\n**";  
 **switch** (ingresar\_enteros("Ingresar la opcion: ")) {  
 **case** **1**:  
 vec.insertar(ingresar\_enteros("**\n\n**Ingresar un numero: "));  
 **break**;  
 **case** **2**:  
 vec.eliminar(ingresar\_enteros("**\n\n**Ingresar el numero a eliminar: "));  
 **break**;  
 **case** **3**:  
 vec.imprimir();  
 **break**;  
 **case** **4**:  
 **return** **0**;  
 **default:**  
 **break**;  
 }  
 } **while** (true);  
}

***Vector.h***

#pragma once  
#include <iostream>  
// Definición del TDA Vector  
**class** **Vector** {  
**private:**  
 **int**\* array; // Puntero al array  
 **int** tamanoActual; // Tamaño actual del vector  
 **int** tamanoVector;  
**public:**  
 Vector(**int** tamInicial) : tamanoActual(**0**) {  
 array = **new** **int**[tamInicial]; // Asigna memoria para el array  
 tamanoVector = tamInicial;  
 }  
 **void** insertar(**int** elemento) {  
 **if** (tamanoActual < tamanoVector) {  
 array[tamanoActual++] = elemento;  
 }  
 **else** {  
 std::cout << "**\n\n**El vector esta lleno**\n**" << std::endl;  
 system("pause");  
 }  
 }  
 **void** eliminar(**int** dato) {  
 **int** indice = -**1**;  
 **for** (**int** i = **0**; i < tamanoActual; ++i) {  
 **if** (array[i] == dato) {  
 indice = i;  
 **break**;  
 }  
 }  
 **if** (indice != -**1**) {  
 **for** (**int** i = indice; i < tamanoActual - **1**; ++i) {  
 array[i] = array[i + **1**];  
 }  
 tamanoActual--; // Decrementar el tamaño del vector  
 }  
 **else** {  
 std::cout << "**\n\n**No se encontro el dato**\n**" << std::endl;  
 system("pause");  
 }  
 }  
  
 **void** imprimir() {  
 std::cout << "**\n\n**Vector: ";  
 **for** (**int** i = **0**; i < tamanoActual; ++i) {  
 std::cout << array[i] << " ";  
 }  
 std::cout << std::endl << std::endl;  
 system("pause");  
 }  
};

***Validacion.h***

#pragma once  
#include <conio.h>  
#include <iostream>  
**int** **borrar**(**char**\* datos, **int**& i) {  
 **if** (i > **0**) {  
 printf("**\b** **\b**");  
 i--;  
 datos[i] = '\0';  
 **return** i;  
 }  
 **return** **0**;  
}  
**int** **ingresar\_enteros**(std::string msj) {  
 **char**\* datos = **new** **char**[**10**];  
 **char** c;  
 **int** i = **0**;  
 std::cout << msj;  
 **while** ((c = \_getch()) != **13** && i < **9**) {  
 **if** ((c >= '0' && c <= '9') || c == **8**) {  
 **if** (c == **8**) {  
 i = borrar(datos, i);  
 }  
 **else** {  
 printf("%c", c);  
 datos[i++] = c;  
 }  
 }  
 }  
 datos[i] = '\0';  
 **return** atoi(datos);  
}

1. **Referencias**

[1] Hernandez, A. (15 de marzo de 2017). Análisis de algoritmos. GoConqr.

https://www.goconqr.com/p/14581124/note\_page/742718