**Universidad Nacional Autónoma de México**

Facultad de Ingeniería

**Laboratorios de computación**

**salas A y B**

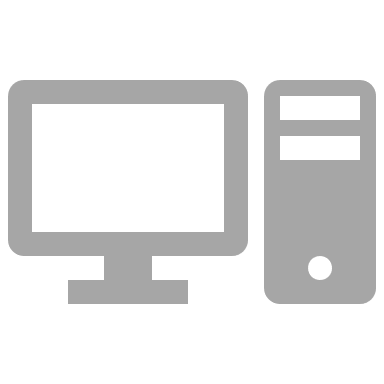
**PROFESOR:** M.I. Marco Antonio Martínez Quintana

**ASIGNATURA:** Estructura de Datos y Algoritmos I

**GRUPO:** 17

**NO DE PRÁCTICA:** 3

**NOMBRE:** Reyes Mendoza Miriam Guadalupe

**SEMESTRE:** 2020-2

**FECHA DE ENTREGA:** 22/02/2020

**OBSERVACIONES:**

**CALIFICACIÓN:**

**TIPO DE DATO ABSTRACTO**

**OBJETIVO**

Utilizarás estructuras en lenguaje C para modelar tipos de dato abstracto e implementarlos en las estructuras de datos lineales.

**INTRODUCCIÓN**

**DATO ABSTRACTO (TDA)**

Es un conjunto de datos u objetos creado de manera personalizada por un programador para un fin específico. Un TDA es una **abstracción** que permite modelar las características de un elemento en particular.

Un tipo de dato abstracto se puede manipular de forma similar a los tipos de datos que están predefinidos dentro del lenguaje de programación, **encapsulando** más información, según se requiera.

La implementación de un tipo de dato abstracto depende directamente del lenguaje de programación que se utilice. En lenguaje C los tipos de dato abstracto se crean mediante las estructuras (struct).

**DATO ABSTRACTO (TDA)**

Una estructura es una colección de una o más variables, de iguales o diferentes tipos, agrupadas bajo un solo nombre, es decir, es un tipo de dato compuesto que permite almacenar un conjunto de datos de diferente tipo (agrupar un grupo de variables relacionadas entre sí) que pueden ser tratadas como una unidad (bajo un mismo nombre). Las estructuras pueden contener tipos de datos simples y tipos de datos compuestos. Los tipos de datos simples (o primitivos) son: carácter, números enteros o números de punto flotante. Los tipos de datos compuestos son: los arreglos y las estructuras.

Por lo tanto, los tipos de datos abstractos (TDA) en lenguaje C se pueden crear a través de una estructura.

Cada ente u objeto es una abstracción de un elemento y, por ende, se puede modelar a través de una estructura en lenguaje C: una película, un video, una pista musical, un documento a imprimir, las armas de un juego, etc.

La sintaxis para crear estructuras en lenguaje C está definida por la palabra reservada **struct**, seguida del nombre de la estructura y, entre llaves, se definen el número y tipo de variables que definan al **nodo** (abstracción), es decir:

**struct nodo {**

**tipoDato elemento1;**

**tipoDato elemento2;**

**…**

**tipoDato elementoN;**

**};**

Para crear una variable de un tipo de dato abstracto, se debe especificar el nombre de la estructura y el nombre de la variable, es decir:

**struct nodo elemento;**

donde elemento es el nombre de la variable con la que se puede acceder a los datos definidos dentro de la estructura.

**CONCEPTOS**

**ENCAPSULAMIENTO**

El encapsulamiento es un mecanismo para desarrollar módulos, en el que se oculta la información privada (no relevante del resto del programa) por medio de una interfaz pública (conocida por el resto del programa), que aísla dicha información.

De este modo, el encapsulamiento se refiere a la capacidad de minimizar el efecto de la modificación de un módulo en el resto de código relacionado. Así, podemos considerar que distintos diseños tendrán distintos grados de encapsulamiento, es decir, un trozo de código tiene mayor grado de encapsulamiento, si los cambios en el implican me dos codificaciones para el código relacionado.

Por lo tanto, el diseño de tipos de datos abstractos se puede considerar la aplicación del encapsulamiento al problema de crear nuevos tipos de datos. Como era de esperar en un lenguaje que permite la programación dirigida por objetos, la idea de encapsular va enfatizada, ofreciendo nuevos mecanismos que facilitan su uso.

**DESARROLLO Y RESULTADOS**

1. **CÓDIGO: NODO PELICULA**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa hace un listado con diferentes datos de una película implementando el uso de una estructura, desde aquí podemos empezar a ver el modelado de una película y la mayoría de su información. Lo primero que hace es crear una estructura llamada película y después se definen sus variables, estas son de tipo carácter y de tipo entero, las de tipo carácter apuntan a una localidad de memoria donde se va a guardar los caracteres del texto (nombre, género y directores) y las de tipo entero van a definir el año y los números de directores todo pertenecientes a la película.

Posteriormente se hace una función que nos va a permitir dirigirnos a otra que realizara la impresión de los datos que se llenaron a lo largo de programa. También se hace otra función que nos va a permitir llenar o escribir los datos definidos en la estructura, definiendo que es lo que se va a leer.

La función principal inicia apuntando a la dirección donde se guardarán los nombres de los directores y declarando un arreglo de 10 nombres diferentes, aunque en este caso solo sean 2 y de los cuales solo se usan dos casillas (0 y 1) para escribir sus nombres, como el ejemplo. Después se llama a las variables definidas en la estructura y llamando a la función de llenar datos la información es manejada ahí y por último lo que se hace es llamar a la última función que imprimirá los datos de nuestra película definidos durante el proceso.

La función de llenar datos película define de nuevo las variables que se van a usar y hace otras estructura ahora llamada movie aquí se accede a la localidad de cada variable y se guarda la información solo en una variable que es movie, aquí mismo se declara una estructura de repetición for esto se hace para que se recorra el arreglo de los directores y vaya guardando sus nombres de acuerdo a la posición en la que estén y al terminar regresa a la estructura movie.

La última función imprime los datos de la película que están guardados en su estructura, primero imprime el nombre de la película, luego el género, el año y los directores. Los primeros tres los saca de la estructura movie accediendo a la variable y los directores los obtiene de la misma manera solo que aquí tiene que recorrer un arreglo para obtener los nombres y la posición en la que están sus nombres.

Al presentar un error este programa, debemos recordar que nunca se declaro una estructura llamada nodo si no que se declaró con el nombre de película, razón por la cual teníamos que cambiar las que llevaban este nombre (nodo) por el nombre de película si no la estructura no existía y por consiguiente el programa no podía hacer nada.

1. **CÓDIGO: PILA PELICULAS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamenteImagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**EXPLICACION**

Este programa crea una pila de manera que nosotros podamos entender como es que estas funcionan y un ejemplo de como podemos implementarlas en C.

Primero podemos observar que se declaran dos constantes que definen el número de películas y el número de directores que vas a guardar en cada una y mostrarlos en pantalla, al cambiar esto nosotros podemos guardar más películas o más directores.

Se empieza definiendo una estructura que, igual que en la anterior se definen arreglos de 20 caracteres para guardar el nombre de la película y el género, pero también el de el director o directores pues dependiendo de tamaño que se definió anteriormente del número de directores puede ser solo 1 o muchos. También se definen dos variables enteras para escribir el año y el número de directores para guardar sus nombres.

Posteriormente se declaran las dos funciones donde se van a llenar los datos y a imprimir, los cuales están apuntando a la estructura donde se localizan los mismos. Se empieza la función principal que realmente es muy corta pues solo llama a las funciones y a la estructura que a final de cuentas son las que tienen toda la información e instrucciones.

La primera función es para llenar los datos del arreglo los primero que hace es declara dos variables que nos van a permitir hacer un arreglo que se recorre mediante un for para pedir los datos de la película y hasta que no sea el tamaño que declaramos en un principio se va a detener y va a dejar de pedirlos. Lo datos que pide los va guardando en movie accediendo a cada variable de la cual se esta dando la información. Por último, en esta parte se dice que el arreglo donde se guardó la información va a ser solo la variable movie.

La última función imprime en pantalla la estructura del arreglo película de acuerdo al número de películas que son, primero hace un ciclo de repetición for para ir imprimiendo los datos de la película pero tenemos que ver que en este caso al arreglo de la película se le recorre en sentido contrario, es decir, en vez de ir recorriendo una posición empieza por la ultima película y luego va a la anterior a ella (esto es una pila) hasta que se llegue a la posición 0 del arreglo y ya no haya más películas que mostrar; empieza imprimiendo el número de película que primero seria la película 2 y luego sus datos correspondientes y para el caso de los directores lo que hace es otro ciclo para ir escribiendo el numero de director que es y su nombre pues cada uno esta escrito en una casilla diferente del arreglo.

**EJERCICIO**

**Imagen que contiene captura de pantalla

Descripción generada automáticamente Imagen que contiene captura de pantalla

Descripción generada automáticamente**

**RESULTADOS**

**Imagen que contiene texto, periódico

Descripción generada automáticamente Imagen que contiene captura de pantalla

Descripción generada automáticamente**

Como se puede observar en el ejemplo anterior, un tipo de dato abstracto permite modelar cualquier elemento (película en este caso) para ser manipulado como una unidad dentro de una estructura de datos. Los elementos (estructuras en lenguaje C) pueden estar compuestos por tipos de datos simples (enteros, caracteres o reales) o por datos compuestos (arreglos y otras estructuras en lenguaje C), haciendo que el modelado de elementos sea tan específico como se requiera.

**CONCLUSIONES**

Esta práctica me sirvió mucho para conocer cómo es que funcionan las pilas, donde el primer elemento que entra es el último en salir pero también para ver que podemos modelar datos de un objeto de la vida real y convertirlos en un programa, lo cual resulta muy útil pues la mayoría de código y diseño de algoritmos que hagamos durante la vida laboral van a ser justamente para obtener o abstraer datos de cosas o acciones que pasan en la vida real y que ocurren a diario porque justamente eso necesitan para un buen y eficiente manejo de la información que se quiera obtener o utilizar. Como hemos visto en clase aplicaciones que actualmente tiene un gran consumo por la sociedad funcionan con base en esto (pilas), un gran ejemplo fue justamente WhatsApp.

A final de cuentas los lenguajes predefinidos se presentan como datos abstractos ya que el usuario sólo se ocupa de entender un manual y muchas veces no se sabe cómo es la implementación interna, un ejemplo, al usar un tipo de dato *int* no tenemos que saber que su representación interna, si no, para que nos va a servir (valores que guarda y operaciones que realiza).

También llegue a la conclusión de que debemos mejorar nuestra capacidad de análisis para reconocer cual es el error en un programa con varias líneas de código puesto que en el primer programa pase aproximadamente 20 minutos tratando de encontrar el error y no fue hasta que no se nos dio una pista que lo vi porque no era en lo que normalmente tenemos un error como en los paréntesis, las comillas o el punto y coma si no que era en la estructura pues en el programa se principal se utilizaba la estructura con el nombre de nodo y en un inicio se definía como una estructura llamada película por eso el programa compilaba porque no se llamaba a ninguna variable.

**BIBLIOGRAFÍA**

* Garrido C. Antonio, Fernández. V. Joaquín. (2006). *Abstracción y Estructuras de Datos en C++* (1ª Ed.). Madrid, España: Delta Publicaciones Universitarias.

**GITHUB:**

https://github.com/miriamyi01/EDA-I