$tarea_1$

ismalazcano99

I. ¿Que son las estructuras de datos?

En ciencias de la computación, una estructura de datos es una forma particular de organizar datos en una computadora para que pueda ser utilizado de manera eficiente.
Diferentes tipos de estructuras de datos son adecuados para diferentes tipos de aplicaciones, y algunos son altamente especializados para tareas específicas.
Las estructuras de datos son un medio para manejar grandes cantidades de datos de manera eficiente para usos tales como grandes bases de datos y servicios de indización de Internet. Por lo general, las estructuras de datos eficientes son clave para diseñar algoritmos eficientes. Algunos métodos formales de diseño y lenguajes de programación destacan las estructuras de datos, en lugar de los algoritmos, como el factor clave de organización en el diseño de software.

Existen numerosos tipos de estructuras de datos, generalmente construidas sobre otras más simples:

Un vector es una serie de elementos en un orden específico, por lo general todos del mismo tipo (si bien los elementos pueden ser de casi cualquier tipo). Se accede a los elementos utilizando un entero como índice para especificar el elemento que se requiere. Las implementaciones típicas asignan palabras de memoria contiguas a los elementos de los arreglos (aunque no siempre es el caso). Los arreglos pueden cambiar de tamaño o tener una longitud fija. Un vector asociativo (también llamado diccionario o mapa) es una variante más flexible que una matriz, en la que se puede añadir y eliminar libremente pares nombre-valor. Una tabla de hash es una implementación usual de un arreglo asociativo. Un registro (también llamado tupla o estructura) es una estructura de datos agregados. Un registro es un valor que contiene otros valores, típicamente en un número fijo y la secuencia y por lo general un índice por nombres. Los elementos de los registros generalmente son llamados campos. Una unión es una estructura de datos que especifica cuál de una serie de tipos de datos permitidos podrá ser almacenada en sus instancias, por ejemplo flotante o entero largo. En contraste con un registro, que se podría definir para contener un flotante y un entero largo, en una unión, sólo hay un valor a la vez. Se asigna suficiente espacio para contener el tipo de datos de cualquiera de los miembros. Un tipo variante (también llamado registro variante o unión discriminada) contiene un campo adicional que indica su tipo actual. Un conjunto es un tipo de datos abstracto que puede almacenar valores específicos, sin orden particular y sin valores duplicados. Un Multiconjunto es un tipo de datos abstracto que puede almacenar valores específicos, sin orden particular. A diferencia de los conjuntos, los multicunjuntos admiten repeticiones. Un grafo es una estructura de datos conectada compuesta por nodos. Cada nodo contiene un valor y una o más referencias a otros nodos. Los grafos pueden utilizarse para representar redes, dado que los nodos pueden referenciarse entre ellos. Las conexiones entre nodos pueden tener dirección, es decir un nodo de partida y uno de llegada. Un árbol es un caso particular de grafo dirigido en el que no se admiten ciclos y existe un camino desde un nodo llamado raíz hasta cada uno de los otros nodos. Una colección de árboles es llamada un bosque. Una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Las clases se utilizan como representación abstracta de conceptos, incluyen campos como los registros y operaciones que pueden consultar el valor de los campos o cambiar sus valores.

II. ¿Para que se usan?

En programación, una estructura de datos puede ser declarada inicialmente escribiendo una palabra reservada, luego un identificador para la estructura y un nombre para cada uno de sus miembros, sin olvidar los tipos de datos que estos representan. Generalmente, cada miembro se separa con algún tipo de operador, carácter o palabra reservada.

En el lenguaje de programación Pauscal, es posible crear una estructura de datos de la forma mencionada. La sintaxis básica es:

Estruct Identificador,
Miembro1:TipoDeDato,
Miembro2:TipoDeDato,
Miembro9:TipoDeDato Para acceder a los miembros de una estructura, primero se debe crear una referencia a esta, generalmente con una variable de tipo; luego se pueden editar y obtener los datos de los miembros libremente.

Estruc Estructura,Miembro1:Entero,Miembro2:Cadena,Miembro3:Byte
Var Variable:Estructura
Variable.Miembro1 = 40000
Variable.Miembro2 = ”Hola Mundo”
Variable.Miembro3 = 255
Mensaje(Variable.Miembro2) ’ Muestra ”Hola Mundo”

III. ¿Que son los tipos de datos abstractos?

Un tipo de dato abstracto (TDA) o tipo abstracto de datos (TAD) es un modelo matemático compuesto por una colección de operaciones definidas sobre un conjunto de datos para el modelo.

Con mucha frecuencia se utilizan los términos TDA y Abstracción de Datos de manera equivalente, y esto es debido a la similitud e interdependencia de ambos. Sin embargo, es importante definir por separado los dos conceptos.

Como ya se mencionó, los Lenguajes de Programación Orientados a Objetos son lenguajes formados por diferentes métodos o funciones y que son llamados en el orden en que el programa lo requiere, o el usuario lo desea. La abstracción de datos consiste en ocultar las características de un objeto y obviarlas, de manera que solamente utilizamos el nombre del objeto en nuestro programa. Esto es similar a una situación de la vida cotidiana. Cuando yo digo la palabra "perro", usted no necesita que yo le diga lo que hace el perro. Usted ya sabe la forma que tiene un perro y también sabe que los perros ladran. De manera que yo abstraigo todas las características de todos los perros en un solo término, al cual llamo "perro". A esto se le llama 'Abstracción' y es un concepto muy útil en la programación, ya que un usuario no necesita mencionar todas las características y funciones de un objeto cada vez que este se utiliza, sino que son declaradas por separado en el programa y simplemente se utiliza el término abstracto ("perro") para mencionarlo.

En el ejemplo anterior, "perro" es un Tipo de Dato Abstracto y todo el proceso de definirlo, implementarlo y mencionarlo es a lo que llamamos Abstracción de Datos.

Vamos a poner un ejemplo real de la programación. Supongamos que en algún Lenguaje de Programación Orientado a Objetos un pequeño programa saca el área de un rectángulo de las dimensiones que un usuario decida. Pensemos también que el usuario probablemente quiera saber el área de varios rectángulos. Sería muy tedioso para el programador definir la multiplicación de 'base' por 'altura' varias veces en el programa, además que limitaría al usuario a sacar un número determinado de áreas. Por ello, el programador puede crear una función denominada 'Área', la cual va a ser llamada el número de veces que sean necesitadas por el usuario y así el programador se evita mucho trabajo, el programa resulta más rápido, más eficiente y de menor longitud. Para lograr esto, se crea el método Área de una manera separada de la interfaz gráfica presentada al usuario y se estipula ahí la operación a realizar, devolviendo el valor de la multiplicación. En el método principal solamente se llama a la función Área y el programa hace el resto.

Al hecho de guardar todas las características y habilidades de un objeto por separado se le llama Encapsulamiento y es también un concepto importante para entender la estructuración de datos. Es frecuente que el Encapsulamiento sea usado como un sinónimo del Ocultación de información, aunque algunos creen que no es así.

IV. Cual es la diferencia entre C y C++

C es un lenguaje libre estandarizado por ISO MUY PEQUEÑO que admite programación estructurada (la de toda la vida), nada más. Útil en programación de microchips, sistemas operativos, drivers y programación web... Poco más.

C++ es un lenguaje libre estandarizado por ISO MUY GRANDE, que admite: Programación estructurada (la de toda la vida), la totalidad de la POO (objetos, herencia simple, herencia múltiple, polimorfismo, upcasting, downcasting, RTTI, interfaces, clases abstractas, clases amigas, operadores, sobrecarga... y mil cosas más avanzadas), programación genérica (plantillas, STL, conceptos de contenedores, estructuras de datos genéricas, ¡b¿metaprogramación¡/b¿... es una programación que no dispone ningún otro lenguaje mayoritario por el momento)... y otras características más avanzadas como los punteros inteligentes, programación lambda, programación "física"... También se usa como programación por eventos (MFC, Qt, Gtk y otras GUIs). Se usa para cualquier cosa, desde sistemas operativos hasta los juegos 3D de última generación pasando por servidores, pasando por las típicas aplicaciones de escritorio o un Office o OpenOffice, un reproductor WinAMP y Windows Media Player, un explorador web (Internet Explorer, Mozilla, Firefox...), un cliente eMule, un cliente Torrent.

C es un invento de Microsoft (lenguaje propietario) que mezcla las características básicas de C++ (no las avanzadas) simplificándolas al estilo Java y ofreciendo un framework. El problema es que es .Net, y deja de ser código nativo/portable. Eso sí, el framework provee bastante facilidad de programación de tareas comunes, al igual que Java. Por ello se genera el debate ¿Java o C? Su funcionalidad viene a ser parecida. .Net es más nativo y Java más virtual.

Aunque C++ es un superconjunto de C, existen algunas diferencias entre los dos. En primer lugar, en C cuando una función no toma parámetros, su prototipo tiene la palabra void. Sin embargo en C++ void no es necesario(opcional).

Prototipo en C: char f1(void);
Prototipo en C++: char f1();

Otra diferencia entre C y C++ es que en un programa de C++ todas las funciones deben estar en forma de prototipo, en C los prototipos se recomiendan, pero son opcionales. También si una función de C++ es declarada para devolver un valor obligatoriamente la sentencia return debe devolver un valor, en C no es necesario que se devuelva.

Otra diferencia es el lugar donde se declaran las variables locales. En C, deben ser declaradas solo al principio del bloque, mientras que en C++ las variables se pueden declarar en cualquier punto. Aunque es conveniente realizarlo siempre al comienzo de la función.