

Organización Lógica de los Datos. Estructuras Estáticas

TEMA 11

ABACUS NT

Oposiciones 2021

Índice

1. Introducción

2. Organización Lógica de los Datos

2.1. Descripción de datos

2.2. Tipos de datos

- 2.2.1. Tipos simples y compuestos
- 2.2.2. Ventajas y desventajas de las estructuras de datos estáticas/dinámicas

3. Estructuras Estáticas

3.1. Matrices

- 3.1.1. Definición y Estructura
- 3.1.2. Tipos de matrices

3.2. Registros

- 3.2.1. Definición
- 3.2.2. Declaración:
- 3.2.3. Estructura de los registros.
- 3.2.4. Factor de Bloqueo
- 3.2.5. Tipos de Registros
- 3.2.6. Registros de longitud fija.
- 3.2.7. Registros de longitud variable
- 3.2.8. Registros de longitud indefinida.
- 3.2.9. Operaciones con registros

3.3. TDA Conjunto

- 3.3.1. Definición/Declaración
- 3.3.2. Operaciones sobre conjuntos

4. Conclusiones

4.1. Relación con el currículo

5. Bibliografía

1. Introducción

La información en la sociedad actual es un aspecto relevante en múltiples ramas del conocimiento; incluyendo la biología genética, la física cuántica, la divulgación científica y por supuesto la informática, Internet y el tratamiento de la información.

Cada día, se intercambian unos 2,5 billones de bytes de datos. Se estima que el 90% de los datos existentes en el mundo actual se han generado en el curso de los últimos dos años. Este diluvio se conoce como **Big Data**.

Sin embargo, este tremendo gigante en el que se apoya la sociedad moderna, requiere de estructuras de datos y programas que los manipulen de forma eficiente.

Según el autor **Y. Daniel Liang**, en su libro **Introduction to Java Programming and Data Structures, Comprehensive Version (2017)**, “*elegir la estructura de datos más adecuada es, junto con el diseño del algoritmo, la clave para desarrollar software con un buen rendimiento*”.

2. Organización Lógica de los Datos

2.1. Descripción de datos

Un dato puede identificarse de forma directa dando su valor, o bien asignándole un nombre con el que vamos a reconocer ese dato.

Hay que ver también si ese valor va a mantener fijo su valor a lo largo del problema o bien se va a mantener constante durante todo el proceso, podemos por tanto encontrar datos literales y asociados a un identificador.

- **Literales:** Un literal es la expresión de un valor de un tipo de dato. Por ejemplo, son literales “nombre”, 5, ‘a’. Son, por tanto, valores fijos.
- **Variables:** Una variable representa un conjunto de **posiciones de memoria** que tienen asociado un nombre (**identificador**) y un **valor** (contenido) de un determinado tipo. Para utilizar una variable normalmente hay que declararla primero, asignándole un nombre y un tipo de dato.
- **Constantes:** Una constante es un dato cuyo valor se declara al principio del programa y no variará a lo largo del algoritmo. Igual que las variables se asocia un nombre al valor de la constante.

2.2. Tipos de datos

2.2.1. Tipos simples y compuestos

Los tipos de datos utilizados en los diferentes lenguajes de programación se pueden clasificar en **tipos simples y en tipos compuestos**.

Los tipos de datos simples o **primitivos no están compuestos de otras estructuras** de datos. Los usados más frecuentemente en casi todos los lenguajes son: **enteros, reales o**

carácter. Los datos simples tienen como característica común que cada variable representa a un elemento.

Los tipos de datos **compuestos o estructurados** de datos están construidos **basándose en tipos de datos simples**. El ejemplo más representativo es la **cadena** de caracteres. Los datos estructurados tienen como característica común que **un identificador puede representar a múltiples datos individuales**, pudiendo estos ser referenciados individualmente. Las estructuras de datos se pueden dividir según su modo de almacenamiento en:

- **Internas**: se almacenan en memoria interna, estas a su vez pueden dividirse en.
- **Externas**: Se almacenan en la memoria externa del ordenador: ficheros y bases de datos.

Las estructuras de datos internas, pueden a su vez, clasificarse en:

- **Estáticas**: El tamaño ocupado en la memoria se define antes de que el programa se ejecute y no puede modificarse durante la ejecución del programa. Están implementadas en casi todos los lenguajes y son los registros, ficheros y matrices. Dependiendo del tipo de dato pueden ser:
 - Homogéneas: todos los subelementos son del mismo tipo: arrays o matrices
 - Heterogéneas: los subelementos contienen datos de distintos tipos: registros
- **Dinámicas**: El tamaño de la estructura se va modificando conforme avanza el programa; es decir, su tamaño cambia en tiempo de ejecución.
 - Lineales: listas, pilas, colas
 - No lineales: árboles y grafos
- **Punteros**: Los tipos de datos puntero a apuntadores almacenan direcciones de memoria que a su vez indican la posición de otras estructuras de datos o funciones del programa.

Muy relacionado está también el concepto de **tipo de dato abstracto** que es un **modelo matemático** compuesto por una colección de **operaciones** definidas sobre un conjunto de datos para el modelo (Por ejemplo, el TDA “Cadena”).

El siguiente esquema muestra una clasificación de los tipos de datos más comunes:

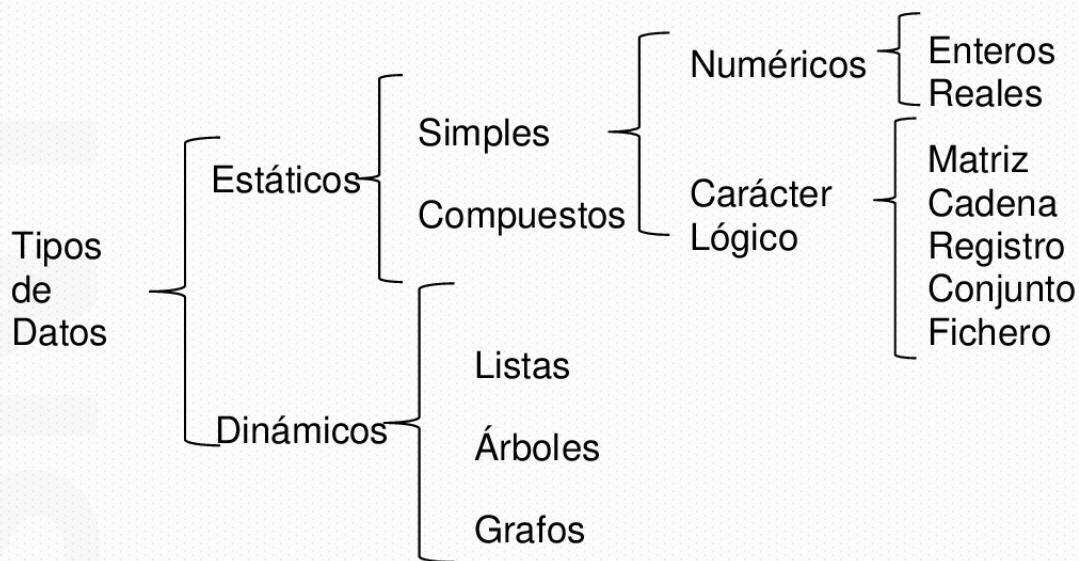


Fig. 1. Tipos de datos.

2.2.2. Ventajas y desventajas de las estructuras de datos estáticas/dinámicas

Según el autor **J. Glenn Brookshear**, en su libro “**Introducción a la computación**” (2012), una ventaja de las estructuras estáticas es que son más **fáciles** de manejar, ya que las estructuras dinámicas tienen que llevar a cabo varias tareas adicionales, como, por ejemplo, buscar bloques de memoria libres, lo que afecta al rendimiento. Por el contrario, las estructuras dinámicas son más **versátiles**, ya que no siempre se conoce previamente el número de elementos que se van a procesar.

3. Estructuras Estáticas

3.1. Matrices

3.1.1. Definición y Estructura

La matriz (array) es una estructura de datos formada por una cantidad fija de datos del mismo tipo, cada uno de los cuales tiene asociado uno o más índices que determinan de forma única la posición del dato en la matriz. Cada índice es un dato de tipo subrango (entero positivo de tamaño limitado). Para cada combinación posible de valores de índices existe sólo un dato del tipo constituyente o elemento del array.

En general, al número de índices del array se le denomina número de dimensiones del array. La dimensión de la formación está dada por los valores máximos de los índices y el número total de elementos es el producto de esos valores máximos.

3.1.2. Tipos de matrices

Vectores

Definición/Declaración

Se denomina vector a una matriz de una dimensión. Un vector se declara de la siguiente manera:

<tipo de dato> Identificador [[tamaño] = {Valor(1), ,Valor(2),...,Valor (tamaño-1)}];

Siendo opcional la inicialización de todos o algunos de sus valores.

Almacenamiento en memoria

Al declarar un vector o en general una matriz, se reserva una cantidad fija de memoria e inalterable durante toda la ejecución del programa, ocupando cada componente un espacio igual al del tipo de dato que aparece en la declaración y estando todas ellas en posiciones contiguas de memoria.

Operaciones

- Asignación: consiste en almacenar un valor en una posición dada.
- Lectura de datos: consiste en recuperar un valor de una posición dada
- Acceso secuencial al vector (recorrido): mediante un bucle recorremos el vector desde la posición inicial hasta la final.
- Actualización
 - Añadir: consiste en añadir un elemento al final del vector si este tiene espacio libre.
 - Borrar: consiste en borrar un elemento del vector, desplazando los posteriores una posición hacia adelante.
 - Insertar: En caso de existir espacio libre al final del vector, consiste en desplazar todos los elementos desde la posición i, una posición hacia atrás y finalmente asignar a la posición i el valor del nuevo elemento.

Matrices n-dimensionales

Definición/Declaración

Una matriz de n dimensiones es un conjunto de elementos, todos del mismo tipo, en el cual el orden de los componentes es significativo, y en el que se necesitan especificar n subíndices para poder identificar a cada elemento de la matriz.

En el caso de que se utilicen dos índices diremos que la matriz es bidimensional, y se usan más de dos, que es multidimensional. Para matrices bidimensionales utilizaremos el término final para referirnos al primer índice y columna para el segundo.

El formato de declaración es el siguiente:

<tipo de dato> Identificador [tamaño1] [[tamaño2] [tamaño3] ... [tamañoN]] ;

Dónde el tamaño 1 corresponde al de la primera dimensión el 2 a la segunda etc.

Almacenamiento en memoria

Al igual que los vectores, todos los elementos de una matriz ocupan posiciones contiguas en memoria.

Como la memoria del ordenador es lineal, una matriz multidimensional debe estar linealizada para su almacenamiento.

Los lenguajes de programación pueden almacenar las matrices en memoria de dos formas:

- Orden de fila mayor
- Orden de columna mayor

El más común en la mayoría de los compiladores es el de orden de fila mayor que consiste en recorrer la matriz iterando los valores desde el valor inicial hasta el final de cada dimensión en orden creciente. Esto se corresponde a una linealización por cada fila, empezando por todos y cada uno de los elementos de la primera, en caso de una matriz bidimensional.

El orden de columna mayor, sin embargo, consiste en linealizar por columnas en caso de una matriz bidimensional, o por cada dimensión empezando por la última en caso de que sean múltiples dimensiones.

Operaciones

Las operaciones sobre una matriz de varias dimensiones son exactamente las mismas que sobre un vector:

- Asignación
- Lectura de datos
- Acceso secuencial (Recorrido) : Es necesario recorrer cada una de las dimensiones con un bucle, anidando tantos bucles como dimensiones tenga la matriz.
- Actualización (Añadir, borrar o insertar) : Igual que para los vectores, pero considerando desplazamientos parciales de la matriz (únicamente para la dimensión afectada) en caso de inserciones y borrados.

TDA Cadena

Definición/Declaración

Una cadena de caracteres (también llamada string) es un tipo de dato abstracto que se define sobre el tipo de dato vector y que tiene asociadas unas operaciones particulares.

Almacenamiento en memoria

Una forma de implementar el TDA Cadena es incorporar un carácter '/0' indicando el final del vector. La mayoría de los compiladores hacen esto de forma transparente. Otra opción podría ser incorporar un valor inicial con la longitud de la misma.

En cualquier caso, es importante no operar con la parte del vector que excede los datos insertados en la cadena y que a la hora de reservar espacio hay que reservar el número de caracteres (bytes) que queramos que tenga la cadena más uno.

Operaciones

Sobre datos de tipo cadena de caracteres se pueden realizar las siguientes operaciones:

- Asignación
- Concatenación: formar una cadena a partir de dos existentes.
- Extracción de subcadena: formando un subcadena a partir de una posición inicial y otra final de otra cadena dada.
- Comparación: Considerando el orden alfabético del primer carácter en el que difieran.
- Longitud: un dato de tipo entero que informa del número de caracteres de la misma.

Aunque la cadena de datos y un vector lineal de caracteres pueden contener la misma información, son tipos de datos distintos, ya que representan objetos distintos y permiten realizar operaciones diferentes.

3.2. Registros

3.2.1. Definición

Un registro, en programación, es un tipo de dato estructurado formado por la unión de varios elementos bajo una misma estructura. Estos elementos pueden ser, o bien datos elementales (entero, real, carácter, ...), o bien otras estructuras de datos. A cada uno de esos elementos se le llama campo.

Los campos dentro del registro aparecerán en el mismo orden y se identificarán por un nombre.

El registro será una unidad de almacenamiento homogénea, dentro de un almacén de información al que llamaremos fichero. Así, tendremos que un fichero es un conjunto de registros afines con las mismas características en cuanto a estructura, significado y tipo de tratamiento.

3.2.2. Declaración:

Como hemos citado anteriormente, un registro viene definido por un conjunto de campos que tendrán, cada uno de ellos, un identificador, un orden y un tipo de dato asociado. La forma de declarar una variable de tipo registro será:

```

<id-registro>: registro{
    campo1: tipo1;
    campo2: tipo2;
    ...
    campoN: tipoN;
}

```

3.2.3. Estructura de los registros.

Todos los registros deben tener un campo que los diferencien del resto. Dicho campo denominado clave, servirá para identificar unos registros de otros.

La información contenida en los registros puede ser de tipo fijo o variable, dependiendo del proceso aplicado al fichero, como veremos más adelante.

En un mismo registro, además, pueden existir distintos niveles de agrupamiento de la información. Cada uno de estos niveles determinará la estructura lógica del registro.

3.2.4. Factor de Bloqueo

Los registros se pueden clasificar en:

- Registros lógicos: Es el conjunto de información identifiable acerca de uno de los elementos a los que se hace referencia en el fichero. Es pues una unidad homogénea de información compuesta de campos de datos que se refieren a un determinado objeto.
- Registros físicos: Se le llama a la unidad de transmisión o almacenamiento entre las distintas memorias del ordenador, ya sean estas externas o internas. También se define como el conjunto de información, que de acuerdo a las posibilidades del ordenador se graba o lee a la vez. Los registro físicos son también denominados bloques. La longitud de estos registros, dependerá del fabricante de la máquina y del sistema operativo utilizado.

Tendremos una relación entre el tamaño del registro lógico y el físico, llamado factor de bloqueo. Cuántos más registros lógicos estén dentro de un registro físico, el número de accesos que se realizarán a soportes de memoria será menor.

3.2.5. Tipos de Registros

Los registros pueden ser de **longitud fija, variable o indefinida**. Tendrán distinto tratamiento atendiendo a su longitud. Hablaremos de geografía fija cuando tengamos el

mismo número de campos en todos los registros. Y, hablaremos de geografía variable, cuando nos refiramos a distinto número de campos.

3.2.6. Registros de longitud fija.

Son aquellos registros en los que la suma de las longitudes de cada campo (en caracteres) es siempre la misma.

Tipos de diseño:

- Con igual número de campos e idéntica longitud de cada campo en cada registro: la longitud del registro lógico es la misma para todos los registros del fichero. Cada registro tiene el mismo número de campos y la longitud de cada uno de esos campos es la misma en todos los registros. Estos campos son de geografía fija y longitud fija.
- Con igual número de campos y distinta longitud de campo en cada registro: en este caso, aunque la longitud de cada campo puede variar, el número total de ellos sigue siendo el mismo y la longitud total del registro también. Sin embargo, la longitud de un campo de un registro puede ser distinta de la longitud del mismo campo de otro registro. Este tipo de registro es de geografía fija y de distinta longitud en sus campos.
- Con distinto número de campos y distinta longitud de cada campo en cada registro: es evidente que se la longitud final del registro tiene que ser fija y el número de campos es variable, la longitud de estos tiene que ser necesariamente distinta. Este diseño de registros no sigue ningún rigor, a excepción de la longitud final, que siempre será fija. Estos registros son de geografía variable y de distinta longitud en sus campos.

3.2.7. Registros de longitud variable

Los registros de longitud variable no responden realmente a lo que su nombre indica, ya que la ocupación real del soporte y de memoria interna siempre será la misma. La variabilidad de la longitud del registro radicará en la mayor o menor ocupación del espacio total destinado a almacenar un registro. La ocupación total siempre oscilará entre un máximo y un mínimo.

Los registros de longitud variable están formados por una parte de longitud fija (identificativo), por unos campos comunes a todos los registros de la misma longitud en cada registro y de una serie de campos de la misma longitud, pero en número variable. El carácter de variabilidad viene especificado por la ocupación de más o menos campos de este tipo.

Este tipo de organización no es el más adecuado, ya que desaprovecha espacio tanto en el soporte como en la memoria interna, debido a que no se ocupa totalmente el espacio reservado para almacenar la información de cada registro. Los campos que se quedan sin información ocupan el mismo espacio que si la tuvieran.

El manejo de estos registros es sencillo, parecido al de los registros de longitud fija.

3.2.8. Registros de longitud indefinida.

Diseñados generalmente para optimizar la ocupación de los soportes de almacenamiento. Su longitud y su estructura son totalmente variables. La ocupación del soporte es óptima, ya que cada registro ocupa exclusivamente lo que tiene.

La longitud de estos registros será un poco mayor que la longitud real de almacenamiento, ya que necesitan unos espacios auxiliares para almacenar unos caracteres de control que indiquen inicio y fin de campo, así como inicio y fin de registro.

Se pueden diseñar de tres maneras:

- Por separadores de campo o banderas: Al final de cada campo se coloca un carácter especial que lo delimita. Para indicar el final lógico del registro utilizamos un carácter especial distinto. Obviamente, ninguno de estos caracteres especiales puede formar parte de ningún campo de datos del registro.
- Mediante indicadores de longitud: se incluyen al inicio de cada campo unos indicadores que especifican la longitud del mismo.
- Mediante máscaras: La máscara es un campo fijo, generalmente ubicado en la primera parte del registro y que indica la ausencia o presencia de campos en el mismo con un cero o un 1 en cada posición de la máscara.

3.2.9. Operaciones con registros

- Asignación. Con los registros se pueden realizar asignaciones del registro completo a una variable de tipo registro, definida con los mismos campos en el mismo orden.
- Lectura. Se puede realizar, al igual que en matrices, la lectura de un campo. Para ello deberemos saber el nombre del registro y el nombre del campo. Sólo podemos acceder a un único campo del registro de forma simultánea
- Escritura. Localizamos dentro de nuestra variable registro cada uno de los campos y les iremos dando valores.

3.3. TDA Conjunto

3.3.1. Definición/Declaración

Un conjunto es una colección de elementos distintos entre sí, y que van a pertenecer al mismo tipo. Este tipo (denominado tipo base) puede ser el propio tipo conjunto o bien un elemento primitivo. Se suele exigir que los elementos del conjunto pertenezcan a tipos ordinales simples (Entero, Carácter, lógico, enumerado y subrango)

Los conjuntos se pueden representar encerrando sus elementos entre llaves {e1,e2,... eN} donde ei representa un elemento del conjunto.

Entre los elementos del conjunto debe existir una relación de orden que no tiene por qué depender del orden que tengan dentro del conjunto y que permite obtener el predecesor y el sucesor de un elemento dado.

3.3.2. Operaciones sobre conjuntos

El TDA conjunto admite las relaciones y operaciones usuales en la definición matemática de conjuntos, esto es:

- Perteneciente a (notación \in) así sí $x \in C$ denota que el elemento x pertenece al conjunto C.
- Incluido en (notación \subseteq), así $A \subseteq C$ denota que cualquier elemento de A está en C, pero no a la inversa.
- Unión. Si A y B son conjuntos, $A \cup B$ tendrá como elementos los que pertenecen sólo a A, los que pertenecen sólo a B y los comunes.
- Intersección: Si A y B son conjuntos, $A \cap B$ estará formado solamente por los elementos comunes a ambos.
- Diferencia: $A - B$ contendrá los elementos de A que no están en B
- Asignación: Asigna a un conjunto sus elementos.

Como en otros tipos de datos, sobre el TDA Conjunto se definen también operaciones sobre sus elementos:

- Comparaciones: Igualdad, menor que, mayor que
- Insertar elemento: Introduce un elemento nuevo en el conjunto
- Suprimir elemento: elimina un elemento del conjunto.

4. Conclusiones

Según el autor **J. Glenn Brookshear**, en su libro “**Introducción a la computación**” (2012), una ventaja de las estructuras estáticas es que son más **fáciles** de manejar, ya que las estructuras dinámicas tienen que llevar a cabo varias tareas adicionales, como, por ejemplo, buscar bloques de memoria libres, lo que afecta al rendimiento. Por el contrario, las estructuras dinámicas son más **versátiles**, ya que no siempre se conoce previamente el número de elementos que se van a procesar.

Programar **eficazmente implica elegir un tipo de estructura adecuada** de datos a la tarea que se desea programar.

Es necesario indicar que la distinción entre estructuras estáticas y dinámicas es sólo aplicable a los lenguajes de programación **compilados**, mientras que, en los lenguajes **interpretados**, como por ejemplo **PHP**, los vectores pueden crecer de forma dinámica, o como en **Java**, el tamaño de los vectores se puede determinar en tiempo de ejecución, ya que en estos lenguajes se implementa un método de reserva dinámica de memoria, aplicable a cualquier tipo de dato compuesto.

4.1. Relación con el currículo

Este tema es aplicado en el aula en los módulos profesionales siguientes, con las atribuciones docentes indicadas (PES/SAI):

Grado Superior

- Lenguajes de Marcas y Sistemas de Gestión de la Información (DAW - DAM –ASIR) (PES)
- Programación (DAW / DAM) (PES)
- Entornos de desarrollo: (DAW / DAM) (PES)

ESO y Bachiller

- Tecnologías de la Información y la Comunicación I (PES)
- Tecnologías de la Información y la Comunicación II (PES)
- Programación y computación (libre configuración autonómica) (PES)

5. Bibliografía

- Y. Daniel Liang. , Introduction to Java Programming and Data Structures, Comprehensive Version. Pearson, 11^a edición. 2017.
- Francisco Javier Moldes Teo. , Java 9. Anaya. 2017.
- J, Glenn Brookshear, Introducción a la computación. Edt. Pearson, 11^º edición. 2012
- Luis Joyanes Aguilar , Fundamentos de programación. Algoritmos, estructuras de datos y objetos., McGraw-Hill, 4^º edición. 2008.
- Langsam, Augenstein y Tanembaum: “Estructuras de Datos con C y C++”, Prentice-Hall 1997
- Prieto A., Lloris A. y Torres J.C.: Introducción a la Informática, McGraw-Hill, 4^a ed 2010

