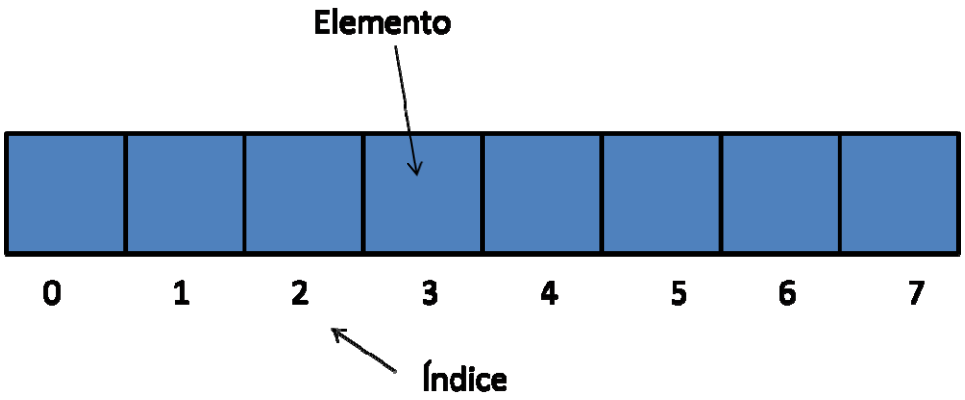
UNIVERSIDAD DISTRITAL
FRANCISCO JOSE DE CALDAS



Arreglos

2013
Transversal de Programación Básica
Proyecto Curricular de Ingeniería de Sistemas

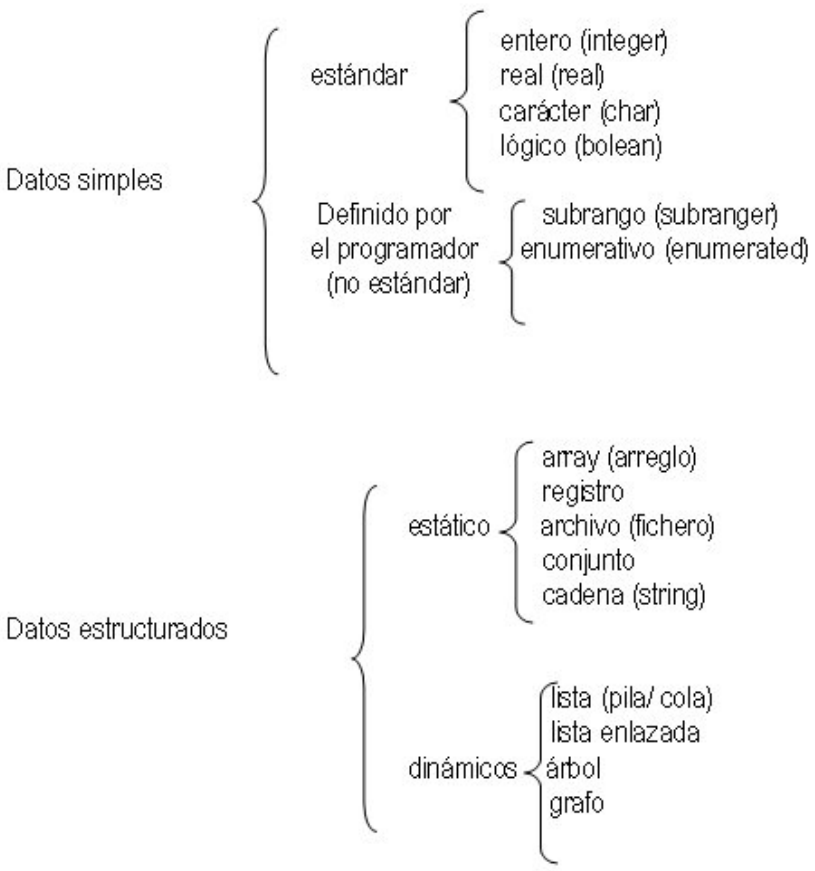
1. Objetivos

- Asimilar el concepto de arreglo.
- Identificar problemas que pueden ser solucionados mediante arreglos.
- Solucionar problemas utilizando arreglos.

2. Introducción

Una estructura de datos es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen en ella. Algunas veces a estas estructuras se les llama tipos de datos.

Una estructura de datos es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen en ella. Las estructuras de datos son muy importantes en los sistemas de computadora. Los tipos de datos más frecuentes utilizados en los diferentes lenguajes de programación son:



Las **estructuras de datos estáticas** son aquellas en las que el tamaño ocupado en la memoria se define antes de la ejecución del programa y no puede modificarse durante esta.

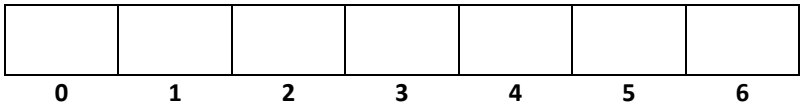
Las **estructuras de datos dinámicas** no tienen las limitaciones y restricciones en el tamaño de memoria de las estructuras estáticas. Mediante el uso de un tipo de dato específico, denominado *puntero,* es posible construir estructuras de datos dinámicas soportadas por la mayoría de los lenguajes de programación.[1]

Una característica importante que diferencia a los datos simples de los estructurados es que para los datos simples cada variable representa un elemento, mientras que en los datos estructurados cada *identificador* (nombre) representa múltiples datos individuales, pudiendo cada uno de ellos ser referenciados individualmente.

Un *arreglo* es un conjunto finito y ordenado de elementos homogéneos. La propiedad ordenado significa que el primer elemento, el segundo, el tercero,..., el enésimo puede ser identificado. La homogeneidad implica que todos los elementos del arreglo son datos del mismo tipo.

3. Definición

Un arreglo es una estructura de datos, o más técnicamente, un espacio de memoria que permite almacenar una colección de elementos, todos del mismo tipo. Conviene imaginar un arreglo como una secuencia contigua de celdas (espacios de memoria), o ***casillas***, en cada una de las cuales se puede guardar un elemento de la colección. Además, es usual dibujarlo como lo ilustra la figura siguiente:



Esta figura representa un arreglo de siete casillas cada una de las cuales se puede utilizar para guardar un dato. La ***dimensión*** o tamaño de un arreglo es el número de casillas que lo conforman. Debe ser claro, entonces, que la figura anterior corresponde a un arreglo de dimensión 7.

Cada una de las casillas de un arreglo tiene asociado un número que la identifica de manera única. A este número se le llama ***índice*** o ***dirección***. En la figura anterior, debajo de cada casilla, aparece su índice. En lenguajes como C, C++ y java, la primera casilla del arreglo tiene índice 0, la segunda tiene índice 1, la tercera índice 2, y así sucesivamente. Es muy importante tener presente que si el arreglo es de dimensión **N**, la última casilla tiene índice **N-1**.

Otras posibles notaciones pueden ser:

[1] Cuando un lenguaje de programación no soporta de estructura de datos dinámicas eventualmente pueden ser simuladas mediante el algoritmo apropiado.

$a_1, a_2, \ldots, a_n$ en matemática y algunos lenguajes (V.B 6.0 y VB.Net)

A(1), A(2),.....,A(I),.....A(N)

A[1], A[2],.....,A[I],.....A[N] en programación (C y Pascal)

Los lenguajes de programación, permiten que el programador declare arreglos de cualquier tipo y prácticamente de cualquier tamaño. En el seudolenguaje, un arreglo se declara usando el siguiente formato o plantilla:

<NOMBRE> : arreglo [<N>] de <TIPO>

En este formato aparecen en mayúsculas y entre los caracteres **<** y **>** los componentes que el programador debe determinar. Así por ejemplo, si se quiere declarar un arreglo con nombre **letras**, de dimensión **15** y que pueda almacenar datos de tipo **caracter**, se debe escribir la siguiente línea.

letras : arreglo [15] de carácter

Volviendo al formato anterior, el programador debe bautizar el arreglo (ponerle un nombre significativo), debe decir cuál es su dimensión, y también debe decir de qué tipo son los elementos que almacenará ese arreglo. Enseguida se dan algunos ejemplos de declaraciones de arreglos.

- Si se necesita guardar las ventas diarias de una tienda durante la última semana, se puede declarar el siguiente arreglo:

ventas : arreglo [7] de real

- Si se quiere guardar las notas que ha sacado un estudiante en los cinco talleres y en los cinco laboratorios del curso de Programación Básica se pueden declarar los siguientes arreglos:

talleres : arreglo [5] de real
laboratorios : arreglo [5] de real

- Si se quiere guardar el valor de las últimas 12 facturas telefónicas de una casa, se puede declarar el siguiente arreglo:

facturasTel : arreglo [12] de real

Los índices se crearon para permitir que el programador se pueda referir, de forma específica, a una cualquiera de las casillas del arreglo, tanto para guardar un dato en esa casilla, como para obtener el dato guardado. Para referirse a una casilla particular de un arreglo se debe seguir el siguiente formato:

<NOMBRE>[<INDICE>]

es decir, se debe escribir el nombre del arreglo seguido por el índice de la casilla entre paréntesis cuadrados.

Ejemplos

A continuación se muestran algunos ejemplos con el objeto de esclarecer la teoría antes presentada

- **Ejemplo Uno**

Para el siguiente ejemplo, suponga que se declara el arreglo **cifras**, de la siguiente manera:

cifras : arreglo [10] de entero

- La siguiente instrucción asigna o guarda el número 100 en la primera casilla de este arreglo:

cifras[0]:= 100

- La siguiente instrucción iterativa guarda 550 en cada una de las últimas 5 casillas de este arreglo:

```
i:=5
MIENTRAS (i<10) HACER
        cifras[i]:= 550
        i:=i+1
FIN-MIENTRAS
```

La siguiente figura muestra el arreglo **cifras** después de ejecutadas las instrucciones de los dos ejemplos anteriores. Las casillas vacías no tienen valores definidos.

100					550	550	550	550	550
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**

- **Ejemplo Dos**

Un ***histograma*** para una colección de datos es una secuencia de parejas de la forma **(d,f)**, donde **d** es un dato y **f** es su frecuencia en la colección. Por ejemplo, suponga que se le pide a 20 personas calificar con las letras **a b c d** y **e** el desempeño del gobierno actual, y que se obtienen las siguientes respuestas: **c b c a b c d e e a b b d c a c c b d a**. El histograma para esta colección de datos se muestra enseguida en dos formas: con números y con asteriscos:

a: 4	**a: ******
b: 5	**b: *******
c: 6	**c: ********
d: 3	**d: *****
e: 2	**e: ****

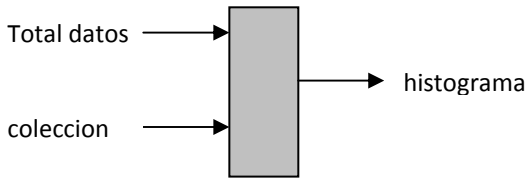
Considere el problema de construir un algoritmo que haga un histograma para una lista de hasta 100 valores, donde cada valor es un número entero comprendido en el intervalo 1 al 5. Las entradas (datos conocidos) para el algoritmo son:

- El número de datos de la colección
- La colección misma

La salida esperada (dato desconocido) es:

- El histograma de la colección

En este problema, los arreglos son útiles para guardar los datos que conforman la colección y también para guardar el histograma. El número de datos de la colección se puede guardar en una variable entera. La siguiente gráfica resume las entradas y las salidas del algoritmo que se



pretende diseñar. Además bautiza todas las variables mencionadas:

Las condiciones iníciales y finales se pueden expresar mediante dos cláusulas: REQUIERE y GARANTIZA, de la siguiente manera:

REQUIERE: El número de datos de la colección debe ser mayor que cero y menor o igual que cien. (100≥**totalDatos>0**)
Cada uno de los elementos de la colección debe ser un número entre 1 y 5.

GARANTIZA: Calcula el histograma asociado a la colección y lo presenta en la pantalla. La frecuencia de cada dato aparece graficado como una secuencia de asteriscos.

Una primera versión del algoritmo solución puede ser simplemente la siguiente:

Inicio
Paso 1. Leer el número de elementos que tiene la colección
Paso 2. Leer los elementos de la colección
Paso 3. Calcular el histograma
Paso 4. Presentar el histograma en la pantalla
Fin

Los pasos 1 y 2 son interacciones con el usuario que permiten capturar los datos de entrada. La versión inicial se puede refinar detallando estos pasos y además, definiendo las variables necesarias para hacerlos:

Procedimiento principal
Variables
i,totalDatos: entero
colección: arreglo [100] de entero

Inicio
escribir("Por favor digite el número de datos de la colección : (inferior o igual a 100)")
leer(totalDatos)
i:=0
mientras (i<totalDatos) hacer
    escribir("Por favor digite el dato :")
    escribir(i+1)
    leer(colección[i])
    i:= i+1
fin-mientras
Paso 3. Calcular el histograma
Paso 4. Presentar el histograma en la pantalla
Fin

La parte nuclear de la solución es el Paso 3 (calcular el histograma). En este problema particular se sabe que todos los datos están entre 1 y 5, lo cual quiere decir que se necesita calcular cinco frecuencias: la del 1, la del 2, etc. Es natural entonces usar un arreglo de 5 casillas para guardar el histograma, de tal manera que en la casilla 0 estará la frecuencia del 1, en la casilla 1 estará la frecuencia del 2, y así sucesivamente. A esta variable se le llamará "histograma" y su declaración es:

histograma: arreglo[5] de entero

Ahora, para calcular las frecuencias se puede hacer lo siguiente:

Paso 3.1: inicializar las cinco casillas del arreglo histograma en cero.
Paso 3.2: Revisar la colección, dato por dato, e ir incrementando la frecuencia asociada a cada uno de ellos. Así, si se encuentra un 5, se debe incrementar en 1 la frecuencia asociada al 5, es decir, se le debe sumar 1 a histograma[4].

En seudo-lenguaje, el cálculo del histograma puede ser:

```
i:=0
mientras (i<5) hacer     // inicializa las frecuencias en 0. Paso 3.1
    histograma[i]:= 0
    i:= i+1
fin-mientras
i:=0
mientras (i<totalDatos) hacer   // calcula las frecuencias. Paso 3.2
    d:= colección[i]-1
    histograma[d]:= histograma[d]+1
```

```
        i:= i+1
fin-mientras
```

Finalmente, una vez calculado el histograma, se debe presentar en la pantalla. Cada dato se debe presentar junto con su frecuencia, pero se requiere que la frecuencia aparezca como una cadena de asteriscos, en vez de como un número. Por ejemplo, si el dato 8 aparece 3 veces en la colección, en la pantalla debe aparecer 8: ***, como una línea del histograma. Esto quiere decir que para escribir la frecuencia de un dato, se requiere un ciclo que escriba tantos asteriscos como sea la frecuencia.

Concluyendo, el paso 4 (presentar el histograma en la pantalla) se puede refinar como se muestra enseguida:

```
i:=0
mientras (i<5) hacer     // este ciclo recorre el histograma
        escribir(i+1)
        escribir(“: “)
        f:= histograma[i]          // f guarda la frecuencia de dato i+1
        j:= 0
        mientras(j<f) hacer        // este ciclo escribe f asteriscos
escribir (‘*’)
                j:= j+1
        fin-mientras
        escribir(salto-de-linea)
        i:= i+1
fin-mientras
```

El algoritmo completo se presenta enseguida. Se han definido algunas constantes para permitir que el programa sea más fácilmente modificable.

Procedimiento principal
Constantes

```
N 5                    / N es el tamaño máximo del histograma
MAXDATOS 100           / MAXDATOS es el tamaño máximo de la colección
```

Variables

```
i,j,totalDatos: entero
coleccion: arreglo [MAXDATOS] de entero
histograma: arreglo [N] de entero
```

Inicio

```
escribir(“Por favor digite el número de datos de la colección : (inferior o igual a
100)”)
leer(totalDatos)
i:=0
mientras (i<totalDatos) hacer
        escribir(“Por favor digite el dato :”)
```

```
            escribir(i+1)
            leer(colección[i])
            i:= i+1
        fin-mientras
        i:=0
        mientras (i<N) hacer     // inicializa las frecuencias en 0. Paso 3.1
            histograma[i]:= 0
            i:= i+1
        fin-mientras
        i:=0
        mientras (i<totalDatos) hacer   // este ciclo calcula las frecuencias. Paso 3.2
            d:= colección[i]-1
            histograma[d]:= histograma[d]+1
            i:= i+1
        fin-mientras
        i:=0
        mientras (i<N) hacer     // este ciclo recorre el histograma
            escribir(i+1)
            escribir(": ")
            f:= histograma[i]          // f guarda la frecuencia de dato i+1
            j:= 0
            mientras(j<f) hacer        // este ciclo escribe f asteriscos
      escribir ('*')
                    j:= j+1
            fin-mientras
            escribir(salto-de-linea)
            i:= i+1
        fin-mientras
       Fin
```

Nota: Ver anexos

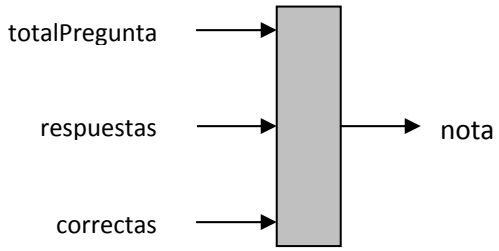
Problemas para desarrollar en clase

1. Suponga que se quiere construir un algoritmo que califique un examen de selección múltiple presentado por un estudiante de Programación Básica. En cada una de las preguntas del examen, el estudiante debió elegir una de cinco opciones, identificadas con las letras A,B,C,D y E. Las entradas (datos conocidos) para el algoritmo son:

   - El número de preguntas que tenía el examen
   - Cada una de las respuestas dadas por el estudiante
   - Las respuestas correctas

   La salida esperada (dato desconocido) es:
   - la nota obtenida. Esta nota corresponde al número de aciertos que tuvo el estudiante.

En este problema, los arreglos son útiles para guardar las respuestas correctas y las opciones elegidas por el estudiante. El número de preguntas del examen se puede guardar en una variable entera, al igual que la nota. Enseguida se muestra la especificación de este problema:



REQUIERE

- El número de preguntas del examen debe ser mayor que cero. (**totalPreguntas>0**).
- Cada una de las respuestas del estudiante debe ser una letra mayúscula que está entre **A** y **E**.
- Cada una de las respuestas correctas debe ser una letra mayúscula que está entre **A** y **E**.

GARANTIZA

- La nota dada por el algoritmo corresponde al total de respuestas acertadas del estudiante

Escriba un algoritmo que cumpla con esta especificación.

2. Escriba un algoritmo que efectúe la normalización de una colección de números reales. Para llevar a cabo esta normalización, se debe en primer lugar encontrar el número mayor de la colección; luego se divide cada número por dicho valor máximo, de forma que los valores resultantes (normalizados) estén comprendidos en el intervalo del 0 al 1.

3. Escriba un algoritmo que sume en binario. Las entradas son dos números (binarios) y la salida es la suma de estos dos valores (también en binario). Por ejemplo, si el usuario digita las cadenas binarias **101** y **1101**, la respuesta dada debe ser **10010**.

Ejercicios para desarrollar en casa

1. Escriba un algoritmo que lea dos arreglos de números enteros ORDENADOS ascendentemente y luego produzca la lista ordenada de la mezcla de los dos. Por ejemplo, si los dos arreglos tienen los números **1 3 6 9 17** y **2 4 10 17**, respectivamente, la lista de números en la pantalla debe ser **1 2 3 4 6 9 10 17 17**.

2. Escriba un algoritmo que lea un arreglo de números enteros, y un número **x**, y escriba en la pantalla todos los índices de las posiciones del arreglo donde está **x**. Por ejemplo, si el arreglo es el que aparece enseguida y **x** es **2**:

1	2	3	100	23	2	2	1

El programa debe escribir: **1 5 6**.

3. Un arreglo de números se llama ***partidario*** si todo número que está en una casilla par (0,2,4,…) es mayor que cualquiera de los números que están en las casillas impares (1,3,5,…). Escriba un algoritmo que lea un arreglo de números enteros y luego, diga si es partidario o no. Por ejemplo, si el arreglo es el siguiente:

100	5	200	1	1000	0	600	50	300	4

El programa debe escribir: **es partidario**.

Anexos

En esta sección se mostrara como es la codificación de la teoría vista en clase, para ello se utilizara como herramienta de codificación: C++.

Codificación en C++ de arreglos y matrices

	Seudocódigo	**C++**
Arreglo	<NOMBRE> : arreglo [<N>] de <TIPO>	<TIPO> <NOMBRE>[<N>];

Ejemplo

```
#include <cstdlib>
#include <iostream>
using namespace std;

int main(int argc, char *argv[])
{
 int const N = 5;                    // N es el tamaño máximo del histograma
 int const MAXDATOS = 100;     // MAXDATOS es el tamaño máximo de la colección;
 int i,j,totalDatos,d,f;
 int coleccion [MAXDATOS];
 int histograma [N];

 cout<<("Por favor digite el número de datos de la colección : (inferior o igual a
100)");
 cin>>(totalDatos);
 i=0;
 while (i<totalDatos){
          cout<<("Por favor digite el dato :");
          cout<<(i+1);
          cin>>(coleccion[i]);
          i= i+1;
```

```
 }
 i=0;
 while (i<N) {     // inicializa las frecuencias en 0. Paso 3.1
           histograma[i]= 0;
           i= i+1;
 }
 i=0;
 while (i<totalDatos) {    // este ciclo calcula las frecuencias. Paso 3.2
           d = coleccion[i]-1;
           histograma[d]= histograma[d]+1;
           i= i+1;
 }
 i=0;
 while (i<N) {     // este ciclo recorre el histograma
           cout<<(i+1);
           cout<<(": ");
           f= histograma[i] ;          // f guarda la frecuencia de dato i+1
           j= 0;
           while(j<f) {        // este ciclo escribe f asteriscos
cout<< ('*');
                     j= j+1;
           }
           cout<<endl;
           i= i+1;
 }

   system("PAUSE");
   return EXIT_SUCCESS;
}
```

Lectura de Profundización

- http://enriquebarrueto0.tripod.com/algoritmos/cap06.pdf
- http://wikipnfi.wikispaces.com/file/view/Arreglo+de+Registro.pdf
- http://aplicaciones.virtual.unal.edu.co/drupal/files/Arreglos%20y%20Matrices%20-%20Programacion%20de%20Computadores.pdf

Imágenes:

Las imágenes utilizadas en este documento fueron tomadas de www.google.com

Fuentes:

- http://www.sites.upiicsa.ipn.mx/polilibros/portal/Polilibros/P_terminados/EstrRepreDat/Files/insercion_nodos.html
- http://lic-toledomarcelo.awardspace.com/archivos/Ejercicios_VB_Arreglos.pdf
- http://cyberprogramacion.zxq.net/pdf/Arrays%20%28Vectores%20y%20Matrices%29%202012%20-%20Luis%20U..pdf
- http://www.fismat.umich.mx/mn1/manual/node6.html
- http://www.herrera.unt.edu.ar/programacion/archivos/ejemplos%20de%20Arreglos.pdf