ESTRUCTURAS DE DATOS Y ALGORITMOS I.

Arreglos -- Arrays.

Concepto.

- Los **arreglos** o **arrays** son un tipo de estructura de datos que almacenan una colección de **tamaño fijo** de elementos del **mismo tipo**.
- En vez de declarar una serie de variables individuales, como podrían ser **num1**, **num2**, **num3**, etc. se declara una sola variable, por ejemplo, **nums** y se usan **nums[0]**, **nums[1]**, **nums[2]** para representar las variables individuales.
- Un elemento específico de un arreglo se accesa mediante un índice.
- El arreglo consiste de localizaciones de memoria contiguas, la dirección más baja corresponde al primer elemento y la más alta al último.

Tipo de Arreglos.

- Existen dos tipos de arreglos:
  - Arreglos simples
    - no existe ningún orden en los elementos del arreglo.
  - Arreglos ordenados
    - los elementos del arreglo se encuentran clasificados u ordenados por uno de los campos del elemento.
    - A este campo se le llama llave.

Nota. En general las llaves deben ser únicas pero en los arreglos es común que se permitan llaves duplicadas.

Declaración de Arreglos .

- Para declarar un arreglo **unidimensional** en C, se especifica el tipo de los elementos y el número de elementos requeridos de la manera siguiente:

tipo nombreDelArreglo[tamañoDelArreglo];

- **tamañoDelArreglo** debe ser un entero constante de cualquier tipo (entero) válido en C.

- Por ejemplo:

double balance[10];

declara una variable de arreglo llamada **balance** suficiente para contener 10 números de tipo double.

Inicialización de Arreglos.

• Se puede inicializar un arreglo en C elemento por elemento:

elem[0] = 1000.0; elem[1] = 2.0; … elem[4] = 50.0; • o bien

en un solo postulado:

double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};

• El postulado anterior asigna al **5°** elemento del arreglo balance el valor 50.0

• Todos los arreglos empiezan con el índice **0** y el último elemento del arreglo es el **tamaño del arreglo menos 1**.

Acceso de Elementos de Arreglos.

- Un elemento de un arreglo se accesa indexando el nombre del arreglo colocando el valor del índice entre corchetes después del nombre del arreglo.

- Por ejemplo:

double sueldo = balance[9];

toma el décimo elemento del arreglo **balance** y lo asigna a la variable sueldo.

- El siguiente ejemplo ilustra el uso de los 3 conceptos, declaración, inicialización y acceso de elementos de un

arreglo.

Ejemplo.

```
#include <stdio.h>
int main () {
   int n[10]; /* n is an array of 10 integers */
   int i,j;
/* initialize elements of array n to 100 + i */
   for ( i = 0; i < 10; i++ ) {
            n[ i ] = i + 100; /* set element at location i to i + 100 */
   }
   /* output each array element's value */
   for (j = 0; j < 10; j++ ) {
       printf("Element[%d] = %d\n", j, n[j] );
   }
```

```
    return 0;
}
```
Ejemplo01.c

Arreglos Multidimensionales.

- El lenguaje C soporta arreglos de cualquier dimensión.

- La forma general de declarar un arreglo de n dimensiones es la siguiente:

type name[size1][size2]...[sizeN];

- Por ejemplo, la siguiente declaración crea un arreglo de enteros de 3 dimensiones:

int tresDim[5][10][4];

continúa…

Arreglos Multidimensionales (2).

- Teóricamente no hay límite para el número de dimensiones; las limitaciones prácticas son el tamaño de la memoria y el compilador utilizado.

- En la práctica rara vez se usan arreglos de más de 3 dimensiones.

Arreglos Bidimensionales.

- La forma más simple y más utilizada de arreglos multidimensionales es el arreglo de dos dimensiones.

- Un arreglo de dos dimensiones es en esencia un arreglo de un arreglo unidimensional.

- Para declarar un arreglo bidimensional se especifica:

tipo nombre [n] [m];

- Donde **tipo** es cualquier tipo de dato válido en C, **nombre** cualquier identificador válido y **n** y **m** constantes de tipo entero.

- Un arreglo bidimensional se puede vizualizar como una tabla de **n** renglones y **m** columnas.

Ejemplo.

- Un arreglo bidimensional **a** de **3** filas y **4** columnas declarado como:

int a [3] [4];

se puede vizualizar como:

Columna 0 Columna 1 Columna 2 Columna 3

Renglón 0 a[0][0] a[0][1] a[0][2] a[0][3] Renglón 1 a[1][0] a[1][1] a[1][2] a[1][3] Renglón 2 a[2][0] a[2][1] a[2][2] a[2][3]

- Cada elemento del arreglo es identificado por un nombre de la forma **a[n][m]** donde **a** es el nombre del arreglo y **n** y **m** son índices que identifican unívocamente cada elemento de **a**.

Inicialización de Arreglos Bidimensionales.

- Los arreglos bidimensionales pueden ser inicializados

especificando listas de valores entre llaves para cada renglón.

- Por ejemplo la inicialización de un arreglo de 3 por 4 se puede especificar como.

int a[3][4] = { {0, 1, 2, 3} , {4, 5, 6, 7} , {8, 9, 10, 11} };

- Las llaves internas, que indican el renglón, son opcionales de manera que la siguiente inicialización es equivalente pero no recomendada porque resulta muy confusa:

int a[3][4] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11};

Acceso de elementos en Arreglos Bidimensionales.

- Los elementos de un arreglo bidimensional se accesan utilizando dos índices, uno para los renglones y otro para las columnas, por ejemplo el postulado:

int val = a[2][3];

toma el **cuarto** elemento del **renglón 2** y lo asigna a la variable **val**.

- Generalmente se utilizan ciclos anidados para manejar los arreglo bidimensionales.

Ejemplo.

```
#include <stdio.h>
int main () {
        /* an array with 5 rows and 2 columns*/
        int a[5][2] = { {0,0}, {1,2}, {2,4}, {3,6},{4,8} };
        int i, j;
        /* output each array element's value */
        for ( i = 0; i < 5; i++ ) {
                for ( j = 0; j < 2; j++ ) {
                        printf("a[%d][%d] = %d\n", i,j, a[i][j] );
                }
        }
        return 0;
}
```

Ejemplo02.c

Arreglos Como Argumentos de Funciones.

Para pasar un arreglo unidimensional como argumento de una función el arreglo se declara como un parámetro formal de la función en una de 3 maneras distintas.

- Forma 1, parámetro como arreglo con tamaño:

  void myFunction(int arreglo[10]) { … }

- Forma 2, parámetro como arreglo sin tamaño:

  void myFunction(int arreglo[ ]) { … }

- Forma 3, parámetro como pointer:

  void myFunction(int *arreglo) { … }

Nota. Se puede usar la misma técnica para arreglo de n dimensiones.

Ejemplo.

```
#include <stdio.h>
double getAverage(int arr[ ], int size) {
        int i;
        double avg;
        double sum;
        double a;
        for (i = 0; i < size; ++i) {
                sum += arr[i];
        }
        avg = sum / size;
        return avg;
}
```

continúa…

Ejemplo (2).

```
int main () {
        /* an int array with 5 elements */
        int balance[5] = {1000, 2, 3, 17, 50};
        double avg;
        /* pass pointer to the array as an argument
        */ avg = getAverage( balance, 5 ) ;
        /* output the returned value */
        printf( "Average value is: %0.2f ", avg );
        return 0;
}
```

Ejemplo03.c

Como Entregar un Arreglo desde una Función.

- El lenguaje C no permite entregar un arreglo como tipo de retorno de una función.
- Sin embargo, se puede entregar un pointer al arreglo especificando el nombre del arreglo sin índice.
- La sintaxis para especificar que se entrega un pointer en una función es la siguiente:

tipo * miFuncion() { … }

- Adicionalmente hay que definir la variable del arreglo como

static, porque C tampoco permite entregar la dirección de una variable local.

Ejemplo.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
/* declaración de la función */
int * getRandom( );
/* función main */
int main () {
   /* a pointer to an int */
   int *p;
   int i;
   p = getRandom();
   for ( i = 0; i < 10; i++ ) {
```

```
        printf( "*(p + %d) : %d\n", i, *(p + i));
    }
    return 0;
```

}continúa…

Ejemplo (2).

```
int * getRandom( ) {
   static int r[10];
   int i;
   srand( (unsigned)time(NULL)); /* random number generator */ for
   ( i = 0; i < 10; ++i) {
       r[i] = rand();
       printf( "r[%d] = %d\n", i, r[i]);
   }
   return r;
```

}

Ejemplo04.c

Los Arreglos son Pointers.

- En realidad, el nombre de un arreglo (la variable que lo representa) no es otra cosa que un pointer o apuntador al primer elemento de un arreglo.

- Por tanto, en la declaración:

  double balance[50];

  balance es un pointer a **&balance[0**] que es la dirección al primer elemento del arreglo balance.

- El siguiente fragmento de un programa asigna **p** como la dirección del primer elemento de balance:

```
double *p;
double balance[10];
p = balance;
```


Los Arreglos son Pointers (2).

- Es legal usar los nombres de los arreglos como pointers y viceversa.

- ***(balance + 4)** es una forma válida de accesar el valor de **balance[4]**.

- Si se almacena la dirección del primer elemento del arreglo en un pointer **p**, se pueden accesar los elementos del arreglo usando ***p, *(p+1), *(p+2),** etc.

Ejemplo.

```
#include <stdio.h>
int main () {
        /* an array with 5 elements */
        double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
        double *p;
```

```
int i;
p = balance;
/* output each array element's value */
printf( "Array values using pointer\n");
for ( i = 0; i < 5; i++ ) {
        printf("*(p + %d) : %f\n", i, *(p + i) );
}
```

continúa…

Ejemplo (2).

```
printf( "Array values using balance as
address\n"); for ( i = 0; i < 5; i++ ) {
        printf("*(balance + %d) : %f\n", i, *(balance + i) );
```

```
        }
        printf( "Array values using balance[index]\n");
        for ( i = 0; i < 5; i++ ) {
                printf("balance[%d] : %f\n", i, balance[i]);
        }
        return 0;
}
```

Ejemplo05.c

Operaciones CRUD en Arreglos

Simples. • Las operaciones CRUD son las

siguientes:

- **C**reate, Insertar.
- **R**etrieve, Buscar, Encontrar, Consultar, Recuperar.
- **U**pdate, Actualizar.
- **D**elete, Borrar.

- En los cuatro casos se debe utilizar un campo de los elementos como llave.
- Los siguientes ejemplos ilustran las operaciones.

Búsqueda en un Arreglo Simple.

```
// Búsqueda en un arreglo simple
#include <stdio.h>
#include <locale.h>
#define LONG_ARR 15
// declaración de la función
int buscar(int num);
// variables globales
int arreglo[LONG_ARR] = {100, 210, 320, 430, 540, 650, 760, 870};
int numElems = 8; // número de elementos en el arreglo
// inicio
int main () {
    setlocale(LC_ALL, ""); // para caracteres del Español int num;
    int result;
    printf("Teclee la llave a buscar: ");
    scanf("%d", &num);
    result = buscar(num);
```

continúa...

Búsqueda en un Arreglo Simple (2).

```
if (result == -1) {
      printf("NO se encontró la llave %d\n ", num);
   } else {
         printf("SÍ se encontró la llave %d\n ", result);
   }
 }
// busca un elemento en un arreglo simple
// entrega el valor encontrado o -1 si no lo encuentra
int buscar(int num) {
        for (int i = 0; i < numElems; i++ ) {
            if(num == arreglo[i]) {
                return num;
            }
        }
       return -1;
   }
}
```
Ejemplo06.c

Inserción en un Arreglo Simple.

```
// Inserción en un arreglo simple
#include <stdio.h>
#include <locale.h>
#define LONG_ARR 15
// declaración de las funciones
int insertar(int num);
void despliega();
// variables globales
int arreglo[LONG_ARR] = {100, 210, 320, 430, 540, 650, 760, 870};
int numElems = 8; // número de elementos en el arreglo // inicio
int main () {
    setlocale(LC_ALL, ""); // para caracteres del Español int num;
    int result;
    printf("arreglo antes: "); despliega();
```

continúa...

Inserción en un Arreglo Simple (2).

```
 printf("Teclee la llave a insertar: ");
    scanf("%d", &num);
   result = insertar(num);
   if (result == -1) {
      printf("NO se pudo insertar, arreglo lleno\n");
   } else {
      printf("SE insertó el elemento con la llave %d\n", result);
   }
    printf("arreglo después: "); despliega();
}
// insertar un elemento en el arreglo
// entrega -1 si el arreglo está lleno
int insertar (int num) {
    // El punto de inserción es en el índice numElems
    if(numElems == LONG_ARR) {
```

continúa...

Inserción en un Arreglo Simple (3) .

```
            return -1;
        } else {
            arreglo[numElems++] = num;
            return num;
        }
}
void despliega() {
    for (int i = 0; i < numElems; i++ ) {
       printf("%d ", arreglo[i]);
             }
             printf("\n");
}
```

Ejemplo07.c

Actualización en un Arreglo Simple.

```
// Actualización en arreglo simple
#include <stdio.h>
#include <locale.h>
#define LONG_ARR 15
// declaración de las funciones
int actualizar(int num, int nuevo);
void despliega();
int buscar(int num);
// variables globales
int arreglo[LONG_ARR] = {100, 210, 320, 430, 540, 650, 760, 870};
int numElems = 8; // número de elementos en el arreglo
// inicio
int main () {
    setlocale(LC_ALL, ""); // para caracteres del Español int num;
    int nuevo;
    int result;
```

continúa...

Actualización en un Arreglo Simple (2).

```
 printf("arreglo antes: ");
 despliega();
 printf("Teclee la llave del elemento a modificar: ");
 scanf("%d", &num);
 printf("Teclee la nueva llave: ");
 scanf("%d", &nuevo);
result = buscar(num);
if (result == -1) { //result es el índice del elemento encontrado printf("NO
   se puede actualizar, no existe la llave %d\n", num);
} else {
    arreglo[result] = nuevo;
    printf("Se actualizó el elemento con la llave %d, la nueva llave es %d\n", num,
nuevo);
}
 printf("arreglo después: ");
 despliega();
}
```

continúa...

Actualización en un Arreglo Simple (3).

```
void despliega() {
    for (int i = 0; i < numElems; i++ ) {
       printf("%d ", arreglo[i]);
                  }
       printf("\n");
 }
// Función buscar, entrega -1 si no lo encuentra, entrega el INDICE cuando lo
encuentra. int buscar(int num) {
    for (int i = 0; i <numElems; i++ ) {
           if(num == arreglo[i]) {
              return i; // entrega el índice NO la llave
           }
                                  }
                                  return -1;
}
```

Ejemplo08.c

Borrado en un Arreglo Simple.

```
// Borrado en un arreglo simple
#include <stdio.h>
#include <locale.h>
#include <stdio.h>
#define LONG_ARR 15
// declaración de las funciones
 int buscar(int num);
 int borrar(int num);
// variables globales
 int arreglo[15] = {100, 210, 320, 430, 540, 650, 760, 870};
 int numElems = 8; // número de elementos en el arreglo
// inicio
 int main () {
```

```
    setlocale(LC_ALL, ""); // para caracteres del Español int num;
    int result;
```

continúa...

Borrado en un Arreglo Simple (2).

```
// busca un elemento en un arreglo simple
// entrega el INDICE del valor encontrado o -1 si no lo encuentra
int buscar(int num) {
        for (int i = 0; i < numElems; i++ ) {
            if(num == arreglo[i]) {
                return i;
            }
                                }
                                return -1;
}
// "borra" el elemento num haciendo un recorrimiento
int borrar(int n) { // n es el índice del valor encontrado. int borrado =
    arreglo[n];
```

```
        for (int i = n ; i < numElems; i++) {
            arreglo[i] = arreglo[i + 1];
         }
```

continúa...

Borrado en un Arreglo Simple (3).

```
 numElems--;
        return borrado;
}
void despliega() {
    for (int i = 0; i < numElems; i++ ) {
       printf("%d ", arreglo[i]);
                }
                printf("\n");
}
```

Ejemplo09.c

Arreglos Ordenados.

- Los arreglos ordenados son arreglos en los que las llaves se encuentran clasificadas por cierto valor.
- Se declaran y se usan de la misma manera que los arreglos simples.
- Sin embargo, las operaciones de búsqueda, inserción y borrado de elementos son radicalmente distintas, puesto que se tiene que tomar en cuenta el valor de la llave para llevar a

cabo las mismas.

- Hay dos tipos de operaciones de búsqueda de elementos:
  - Búsqueda Lineal (igual a la de arreglo simples).
  - Búsqueda Binaria (mucho más rápida).

Algoritmo de Búsqueda Binaria.

- El algoritmo de búsqueda binaria es un algoritmo que encuentra un valor determinado en un arreglo ordenado de una manera mucho más rápida.
- El algoritmo compara el valor buscado con el elemento central del arreglo. Si no son iguales, la mitad en donde no está el buscado, es eliminada y la búsqueda continua en la otra mitad.

- De nuevo se considera la mitad (de la mitad) y se compara el valor central con el valor buscado.
- Se repite el procedimiento hasta que se encuentra el valor buscado. Si la búsqueda termina con la siguiente mitad vacía, significa que el valor buscado no está en el arreglo.
- El algoritmo de búsqueda binaria es O(log N) mientras que el de búsqueda lineal es O(N).

Pseudocódigo de Búsqueda Binaria.

```
Definir variables limInferior, limSuperior y
valorCentral limInferior = 0;
limSuperior = nElems – 1 // numElems es el número de elems en el arreglo
while(true) {
 valorCentral = (limInferior + limSuperior)/2
 if(arreglo[valorCentral] == buscado)
```

```
            return buscado // se encontró
 else if(limInferior > limSuperior)
            return -1; // NO se encontró
          else // acorta el rango
             if(arreglo[valorCentral] < buscado)
                limInferior = valorCentral + 1; // está en mitad superior
else limSuperior = valorCentral - 1; // está en mitad inferior }
```


Otras Operaciones de Arreglos Ordenados.

Insertar.

• Buscar punto de inserción (con búsqueda binaria o lineal). • Recorrer elementos superiores al punto de inserción hacia el final del arreglo.

- Insertar el nuevo elemento en punto de inserción + 1.

Borrar.

- Buscar el elemento a borrar (con búsqueda binaria o lineal).
- Si no se encuentra, terminar.
- Recorrer elementos a partir del encontrado hacia el inicio del arreglo.

Otras Operaciones de Arreglos Ordenados (2).

Actualizar.

- Buscar el elemento a actualizar (con búsqueda binaria o lineal).
- Si no se encuentra, terminar.
- Si se encontró, borrar el anterior e insertar el nuevo.

Ejemplo de Arreglo Ordenado.

// Arreglos Ordenados

```
#include <stdio.h>
#include <locale.h>
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX_ARR 15
// declaración de las funciones
int insertar(int dato);
int busquedaLineal(int dato);
int busquedaBinaria(int dato);
int actualizar(int dato, int nuevo);
int borrar(int dato);
void crearArreglo();
void desplegarArreglo();
// variables globales
int arreglo[MAX_ARR] ; // tamaño máximo del arreglo
int numElems = 0; // número de elementos en el arreglo
```

continúa...

Ejemplo de Arreglo Ordenado (2).

```
// Función Main
int main() {
    setlocale(LC_ALL, ""); // para caracteres del Español int dato,
```

```
    nuevo, resultado;
    crearArreglo(); // crear un arreglo de prueba desplegarArreglo(); // y
    desplegarlo
    printf("Teclee el valor del elemento a encontrar en forma lineal\n");
    scanf("%d", &dato);
    resultado = busquedaLineal(dato);
    if(resultado == -1) { // regresa -1 si no se encuentra printf("No se encontró %d
        en la lista.\n", dato);
    } else {
        printf("Se encontró %d en el arreglo.\n", dato);
    }
    desplegarArreglo();
```

continúa...

Ejemplo de Arreglo Ordenado (3).

```
printf("Teclee valor del elemento a encontrar en forma binaria.\n");
    scanf("%d", &dato);
    resultado = busquedaBinaria(dato);
```

```
    if(resultado == -1) { // regresa -1 si no se encuentra printf("No se encontró %d
        en la lista.\n", dato);
    } else {
        printf("Se encontró la llave %d.\n", dato);
    }
    desplegarArreglo();
    printf("Teclee el valor del elemento a insertar\n");
    scanf("%d", &dato);
    insertar(dato);
    desplegarArreglo();
    printf("Teclee valor del elemento a actualizar.\n");
    scanf("%d", &dato);
    printf("Teclee el nuevo valor del elemento.\n");
    scanf("%d", &nuevo);
```

continúa...

Ejemplo de Arreglo Ordenado (4).

```
resultado =actualizar(dato, nuevo);
    if(resultado == -1) { // actualizar regresa -1 si no se encuentra printf("No se
        encontró %d en la lista.\n", dato);
```

```
    } else {
        printf("Se encontró %d en la lista y se cambio por %d.\n", dato, nuevo);
    }
    desplegarArreglo();
    printf("Teclee valor del elemento a borrar.\n");
    scanf("%d", &dato);
    resultado = borrar(dato);
    if(resultado == -1) { // borrar regresa -1 si no se encuentra printf("No se
    encontró %d en el arreglo, no se puede borrar.\n", dato); } else {
        printf("Se encontró la llave %d y fue borrado el elemento.\n", dato);
    }
    desplegarArreglo();
 return 0;
} // fin de main
```

continúa...

Ejemplo de Arreglo Ordenado (5).

```
// Función Insertar
int insertar(int dato) { // insertar en el arreglo
    if(numElems == MAX_ARR) {
```

```
    return -2; // arreglo lleno
}
int j;
for(j=0; j<numElems; j++) { // encuentra punto de inserción (en forma lineal)
    if(arreglo[j] > dato) { // se encontró
        break;
    }
}
for(int k=numElems; k > j; k--) { // recorre hacia el final del arreglo
    arreglo[k] = arreglo[k-1];
}
arreglo[j] = dato; // inserción
numElems++; // incrementa número de elementos }
```

continúa...

Ejemplo de Arreglo Ordenado (6).

// Función Búsqueda Lineal

```
int busquedaLineal(int dato) { // búsqueda lineal
    int i;
    for(i = 0; i < numElems ; i++) {
        if(arreglo[i] == dato) {
            return dato;
        } else {
            continue;
        }
    }
    return -1;
}
```

continúa...

Ejemplo de Arreglo Ordenado (7).

// Función Búsqueda Binaria

```
int busquedaBinaria(int dato) { // búsqueda binaria int limInferior = 0;
```

```
 int limSuperior = numElems - 1;
 int valorCentral;
 while(true) {
    valorCentral = (limInferior + limSuperior ) / 2;
    if(arreglo[valorCentral] == dato) {
       return valorCentral; // se encontró
    } else if(limInferior > limSuperior) {
       return -1; // NO se encontró
    }
  if(arreglo[valorCentral] < dato) { // divide el rango
       limInferior = valorCentral + 1; // está en la mitad superior } else {
       limSuperior = valorCentral - 1; // está en la mitad inferior }
 }
```

} continúa...

Ejemplo de Arreglo Ordenado (8).

// Función Actualizar

```
int actualizar(int dato, int nuevo) {
   int i;
```

```
    for(i = 0; i < numElems ; i++) { // búsqueda
        if(arreglo[i] == dato) { // se encontró
            break;
        }
    }
    if(i == numElems) { // no se encontró
        return -1;
    }
    borrar(dato); // borra el anterior
    insertar(nuevo); // inserta el nuevo
}
```

continúa...

Ejemplo de Arreglo Ordenado (9).

```
// Función Borrar
int borrar(int dato) {
    int i;
    for(i = 0; i < numElems ; i++) { // búsqueda
```

```
        if(arreglo[i] == dato) {
            break;
        }
    }
    if(i == numElems) { // no se encontró
        return -1;
    }
    for(int k=i; k<numElems; k++) { // mover elementos mayores hacia el inicio
        arreglo[k] = arreglo[k+1];
    }
    numElems--; // decrementa número de elementos return dato;
}
```

continúa...

Ejemplo de Arreglo Ordenado (10).

```
// Función Crear Arreglo

void crearArreglo() {
```

```
    numElems = 0;
    arreglo[0] = 110; numElems ++;
    arreglo[1] = 220; numElems ++;
    arreglo[2] = 330; numElems ++;
    arreglo[3] = 440; numElems ++;
    arreglo[4] = 550; numElems ++;
    arreglo[5] = 660; numElems ++;
    arreglo[6] = 770; numElems ++;
    arreglo[7] = 880; numElems ++;
    return;
}
```

continúa...

Ejemplo de Arreglo Ordenado (11).

Función Desplegar Arreglo

```
void desplegarArreglo() {
    int i;
    printf("[");
    for(i = 0; i < numElems - 1; i++) {
        printf("%d, ", arreglo[i]);
    }
    printf("%d]\n", arreglo[i]);
    return;
}
```