Tipos de datos estructurados

Arreglos unidimensionales, bidimensionales y cadenas de caracteres

Estructuras de Información

- Una estructura de datos o de información es una colección de datos que pueden ser caracterizados por su organización y las operaciones que se definen sobre ellos.

- Las estructuras de datos se utilizan generalmente para procesar una colección de valores que están relacionados entre sí por algún método.

- Las estructuras de datos básicas que soportan la mayoría de los lenguajes de programación son las *estructuras estáticas*.

Estructuras Estáticas

- Una estructura de datos estática es aquella en la que el tamaño ocupado en memoria se define antes de que el programa se ejecute y no puede modificarse dicho tamaño durante la ejecución del programa.

- Entre las estructuras de datos estáticas se encuentran los arreglos (vectores y matrices), registros, archivos y cadenas de caracteres.

Arreglos

- Un arreglo es un conjunto finito y ordenado de elementos homogéneos.
  - Ordenado: cada elemento puede ser identificado de manera individual.
  - Homogéneo: todos los elementos son del mismo tipo de dato.

Arreglos

- De esta manera se puede decir que un arreglo es un conjunto de elementos del mismo tipo agrupados en una sola variable, donde:
  - Se usa un índice para ingresar a cada elemento en particular.
  - Los elementos del arreglo se almacenan en espacios consecutivos en memoria.

Arreglos

- Particularidades y consideraciones:
  - Los arreglos ó conjuntos de datos ordenados (arrays) recolectan variables del mismo tipo, guardándolas en forma secuencial en la memoria.
  - La cantidad máxima de variables que pueden albergar está sólo limitada por la cantidad de memoria disponible.
  - El tipo de las variables involucradas puede ser cualquiera de los ya vistos, con la única restricción de que todos los componentes de un arreglo deben ser del mismo tipo.

Arreglos

- Por sí mismo, el nombre del arreglo apunta a la dirección del primer elemento del arreglo.
- Se puede referenciar a cada elemento, en forma individual, tal como se ha hecho con las variables anteriormente.
- El subíndice designa la posición del elemento en el vector.
- El tamaño o cantidad de elementos en un arreglo siempre debe ser constante.

Arreglos Unidimensionales

- El arreglo unidimensional o vector es el tipo de arreglo más simple.
  - Sintaxis:

```
tipo_dato  identificador[tamaño];

o bien:

tipo_dato  nombre_variable[cantidad de elementos];
```

Arreglos Unidimensionales

- Ejemplos:

```
int variable1[8];
char nombre_completo[50];
float calificaciones[10];
double cantidades[30];
```

Arreglos Unidimensionales

```
int cantidad = 1000;
int numeros[cantidad]; /* ERROR */
```

```
#define CANTIDAD 1000
…
int numeros[CANTIDAD]; /* CORRECTO */
…
```

Arreglos Unidimensionales

- Operaciones: las operaciones que se pueden realizar con los vectores son:
  - inicialización, asignación, lectura, escritura, recorrido, actualización (insertar, borrar, modificar), ordenación, búsqueda, etc.

Arreglos Unidimensionales

- La inicialización de un arreglo local, puede realizarse en su declaración, dando una lista de valores iniciales:

```
int numeros[8] = {2, 6, 5, -2, 3, 6, 8, 9};
```

- Obsérvese que la lista está delimitada por llaves. Otra posibilidad, sólo válida cuando se inicializan todos los elementos del arreglo, es escribir:

```
int numeros[] = {-3, -2, -1, 0, 1, 2, 3, 4, 5, 6};
```

- donde, se obvia la declaración de la cantidad de elementos, ya que está implícita en la lista de valores constantes.

Arreglos Unidimensionales

- También se puede inicializar parcialmente un arreglo, por ejemplo:

```
int numeros[10] = {1, 1, 1};
```

- en éste caso los tres primeros elementos del mismo valdrán 1, y los restantes 0.

- Se puede usar también un ciclo para inicializar todos los elementos de un arreglo a un valor determinado.

```
int arreglo[10];
for (i = 0; i < 10; i++)
    arreglo[i] = -1;
```

Arreglos Unidimensionales

- **Asignación**
  - identificador[posición] = valor;

  arreglo[3] = 8;

- **Lectura**

  scanf("%d",&arreglo[6]);

- **Escritura**

  printf("El valor en la posición 6 es: %d",arreglo[6]);

Arreglos Unidimensionales

```
int numeros[10];

    for(int i = 0; i < 10; i++)
    {
        numeros[i] = i;
        printf("%d\n",numeros[i]);
    }
```

Arreglos Bidimensionales

- Un arreglo bidimensional o matriz se puede considerar como un vector de vectores. Es decir, un conjunto de elementos homogéneos y ordenados en el que se necesita especificar dos subíndices para poder identificar cada elemento del arreglo.

Arreglos Bidimensionales

- Sintaxis:

  tipo_dato identificador[filas][columnas];

  o bien:

  tipo_dato nombre_variable[filas][columnas];

Arreglos Bidimensionales

- Por ejemplo, si se declara:

  int matriz[3][4];

  - esquemáticamente la disposición espacial de los elementos sería:

columnas:	0	1	2	3	
filas: 0	[0][0]	[0][1]	[0][2]	[0][3]	matriz[0]
1	[1][0]	[1][1]	[1][2]	[1] [3]	matriz [1]
2	[2][0]	[2][1]	[2][2]	[2] [3]	matriz [2]

Arreglos Bidimensionales

- Operaciones: Las operaciones que se pueden realizar con matrices son:
  - Inicialización, asignación, lectura, escritura, recorrido, actualización (insertar, borrar, modificar), ordenación, búsqueda, etc.
  - La inicialización de una matriz local, puede realizarse en su declaración, dando una lista de valores iniciales:

```
int matriz[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8 , 9}};
```

Arreglos Bidimensionales

- También se puede inicializar parcialmente una matriz, por ejemplo:

```
int matriz[3][10] = {{1, 1, 1}, {2, 2, 2}, {3, 3, 3}};
```

- en éste caso los tres primeros elementos de cada fila son inicializados y los restantes valdrán 0.

- Se pueden usar también los ciclos para inicializar todos los elementos de una matriz a un valor determinado.

```
int matriz[2][2];
for (i = 0; i < 2; i++)
    for (j = 0; j < 2; j++)
        matriz[i][j] = -1;
```

Arreglos Bidimensionales

- **Asignación**
  - identificador[fila][columna] = valor;

arreglo[1][3] = 8;

- **Lectura**

scanf("%d",&arreglo[3][2]);

- **Escritura**

printf("El valor en la posición 2,3 es: %d",arreglo[2][3]);

Arreglos Bidimensionales

```
int matriz[3][4];
    int cantidad = 1;
    for(int i = 0; i < 3; i++)
        for(int j = 0; j < 4; j++)
        {
            matriz[i][j] = cantidad;
            printf("%d",matriz[i][j]);
            cantidad++;
        }
```