2.7 Tipos de Datos Avanzados

MATRICES

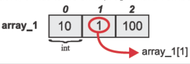
Una matriz es una lista de elementos del mismo tipo colocados en localidades de memoria contiguas. Cada elemento es referenciado por un índice. Para declarar una matriz, es necesario especificar el tipo de sus elementos (denominado tipo de matriz), su nombre y el número de sus elementos encerrados entre corchetes. Todos los elementos de una matriz tienen el mismo tipo.

```
tipo_de_matriz nombre_de_matriz [nº_de_elementos];
```

Los elementos de una matriz se identifican por su posición. En C, el índice va desde 0 (el primer elemento de una matriz) a N-1 (N es el número de elementos contenidos en una matriz). El compilador tiene que "saber" cuántas localidades de memoria debe alojar al declarar una matriz. El tamaño de una matiz no puede ser una variable. Por eso, se pueden utilizar dos métodos:

```
// método 1
int display [3]; // Declaración de la matriz display capaz de contener 3 enteros
// método 2
const DIGITOS = 5;
char Matriz_nueva[DIGITOS]; // Declaración de la matriz Matriz_nueva
                            // capaz de contener 5 enteros
```

Una matriz se puede inicializar a la vez que se declara, o más tarde en el programa. En ambos casos, este paso se realiza al utilizar llaves:




```
int array_1[3] = {10,1,100};
```

Para leer o modificar un elemento de matriz del ejemplo anterior, basta con introducir su índice encerrado entre corchetes:

```
/* Se supone que a ha sido declarado anteriormente como un entero */

a = array_1[0]; // A la variable a se le asigna el valor del miembro de matriz

// con indice 0 (a = 10)
array_1[2] = 20; // Miembro de matriz array_1[2] es modificado (nuevo valor es 20)
```

El siguiente programa cambia el orden de los elementos de una matriz. Note que el índice se puede expresar mediante variables y operaciones básicas.

```
void main() {
  const MUESTRAS_DE_AGUA = 4; // Valor de la constante MUESTRAS_DE_AGUA es 4
  int i, temp; // Variables i y temp son de tipo int
  int profunidad_de_sonda [MUESTRAS_DE_AGUA] = {24,25,1,1987};// Todos

  // los miembros de la matriz profundidad
  // de sonda son de tipo int

  for(i=0;i<(MUESTRAS_DE_AGUA/2);i++){ // Bucle se ejecuta 2 veces
    temp = profundiad_de_sonda [i];    // temp se utiliza para guardar un valor
                                       // temporalmente
    profundiad_de_sonda [i] = profundiad_de_sonda [MUESTRAS_DE_AGUA-1-i];
    profundiad_de_sonda [MUESTRAS_DE_AGUA-1-i] = temp;
  }

  // Aquí tenemos: profundidad_de_sonda [MUESTRAS_DE_AGUA] = {1987,1,25,24}
}
```

MATRICES BIDIMENSIONALES

Aparte de las matrices unidimensionales que se pueden interpretar como una lista de valores, el lenguaje C le permite declarar matrices multidimensionales. En esta parte vamos a describir sólo las matrices bidimensionales, también denominadas tablas o matrices. Una matriz bidimensional se declara al especificar el tipo de dato de matriz, el nombre de matriz y el tamaño de cada dimensión.

```
tipo_de_matriz nombre_de_matriz [número_de_filas] [número_de_columnas];
```

En la declaración de esta matriz número_de_filas y número_de_columnas representan el número de filas y columnas en las que consiste una tabla, respectivamente. Vea la siguiente matriz bidimensional:

```
int Tabla [3][4]; // Tabla se define de modo que tenga 3 filas y 4 columnas
```

Esta matriz se puede representar en la forma de una tabla.

tabla[0][0]	tabla[0][1]	tabla[0][2]	tabla[0][3]
tabla[1][0]	tabla[1][1]	tabla[1][2]	tabla[1][3]
tabla[2][0]	tabla[2][1]	tabla[2][2]	tabla[2][3]

Similar a las matrices unidimesionales, es posible asignar los valores a los elementos de una tabla en la línea de declaración. La asignación debe ser realizada línea a línea como en el siguiente ejemplo. Como hemos visto anteriormente, esta matriz tiene dos filas y tres columnas:

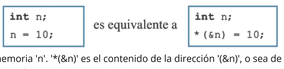
```
int Tabla [2][3]= { {3,42,1},{7,7,19} };
```

3	42	1
7	7	19

La matriz anterior se puede representar también en la forma de una tabla de valores:

PUNTEROS

Un puntero es una variable destinada a recibir una dirección. Un puntero "apunta" a una localidad de memoria, referenciada por una dirección. En C, la dirección de un objeto se puede obtener por medio un operador unitario &. Para acceder al contenido de la memoria en una dirección específica (también llamado objeto apuntado), se utiliza un operador de indirección (*).



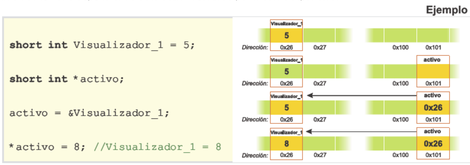
'&n' es la dirección de la localidad de memoria 'n'. '*(&n)' es el contenido de la dirección '(&n)', o sea de 'n'. Para declarar un puntero, se debe que especificar el tipo de la variable apuntada:

```
tipo_de_variable *puntero;
```

En esta etapa, el puntero mi_puntero apunta al valor almacenado en esta localidad de memoria, o sea, a un valor desconocido. Así que, una inicialización es muy recomendable:

```
puntero = &variable;
```

Ahora, puntero contiene la dirección de variable. Para acceder al contenido de la variable apuntada, debe utilizar '*'. El siguiente ejemplo muestra el contenido de memoria dependiendo de la acción realizada por medio del puntero.



Los punteros son muy útiles para manejar las matrices. En este caso, un puntero se utilizará para apuntar al primer elemento de una matriz. Debido al hecho de que es posible realizar operaciones básicas sobre los punteros (aritmética de punteros), es fácil manejar los elementos de una matriz. Fijese en la diferencia entre '*v+1' y '*(v+1)' en el siguiente ejemplo:

```
short int voltio[3] = {0,5,10};
short int *v;
v = &(voltio[0]);  // v contiene la dirección de voltio[0]
*(v+1) = 2;        // voltio[1] = 2
voltio[2] = *v+1;  // tab[2] = 1 (tab[0] + 1)
*(v+2) = *(v+1);   // voltio[2] = 2
v++;               // v contiene la dirección de voltio[1]
*v = 1;            // voltio[1] = 1
```

- Los punteros también pueden ser declarados con el prefijo 'const'. En este caso, su valor no puede ser modificado después de la inicialización, similar a una constante.
- A diferencia de C, el mikroC no admite alojamiento dinámico.

ESTRUCTURAS

Ya hemos visto cómo agrupar los elementos dentro de matrices. No obstante, al utilizar este método todos los elementos deben ser del mismo tipo. Al utilizar estructuras, es posible agrupar diferentes tipos de variables bajo el mismo nombre. Las variables dentro de una estructura se le denominan los miembros de la estructura. Las estructuras de datos se declaran al utilizar la siguiente sintaxis:

```
struct nombre_de_estructura {
  tipo1_de_miembro1 miembro1;
  tipo2_de_miembro2 miembro2;
  tipo3_de_miembro3 miembro3;
  ...
};
```

No es posible inicializar variables dentro de la declaración de la estructura de datos:

```
struct generador {
  int voltaje;
  char corriente;
};
```

Entonces, podrá definir los objetos denominados 'turbina' en el código. A cada uno de estos tres objetos (turbinas) se le asignan las variables 'corriente' y 'voltaje'.

```
struct generadores turbina_1, turbina_2, turbina_3;
```

Para acceder a las variables, es preciso utilizar el operador '.'

```
turbina_3.voltaje = 150;
turbina_3.corriente = 12;
```

Por supuesto, igual que al utilizar los punteros, todavía se le permite realizar operaciones por medio de operadores y sentencias definidos en las partes anteriores. Si está familiarizado con el lenguaje C, recuerde que mikroC no admite la inicialización de los miembros de estructura por medio de las llaves. Por ejemplo, 'conjunto_1 ={15,'m'};' devuelve un error en mikroC.