Arreglos

miércoles, 15 de septiembre de 2021 07:17 a. m.

Los arreglos son estructuras de datos consistentes en un conjunto de datos del mismo tipo. Tienen un tamaño que refleja la cantidad de elementos del mismo tipo que pueden almacenar.

Los arreglos son entidades estáticas debido a que se declaran con cierto tamaño inicial y que se conserva todo el tiempo de ejecución.

Cada elemento del arreglo se almacena de forma consecutiva en la memoria.

0x3 0x4
1, 2, 3, 4, 5

int variable1=1 0x304
Int variable2=2 0x289
Int variable3=3 0x2334899

Sintaxis de declaración de arreglos:

<tipo de dato> identificador[<<Tamaño>>];

Observar que en la declaración se especifica: tipo de dato de los elementos, número de elementos y el nombre del arreglo.

Un arreglo consta de posiciones de memoria consecutivas. La dirección de memoria más baja corresponde al primer elemento y la más alta corresponde al último elemento. Para acceder a un elemento en particular se utiliza un índice.

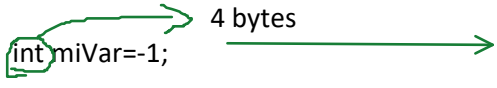
En el lenguaje C, el primer elemento se encuentra en el índice 0. Y en un arreglo de tamaño N, el último elemento se encuentra en el índice N-1.

Ejemplos de declaración de arreglos:

```
//arreglo de enteros para almacenar 20 datos
int arreglo[20];

//arreglo de caracteres para almacenar las letras del nombre Ramon
char nombrePersona[5];

//arreglo de 100 flotantes
float valoresEnDecimal[100];
```

Los ejemplos que se acaban de expresar son de una sola dimensión, también llamados vectores o arreglos unidimensionales que en memoria se podrían visualizar de la siguiente forma:

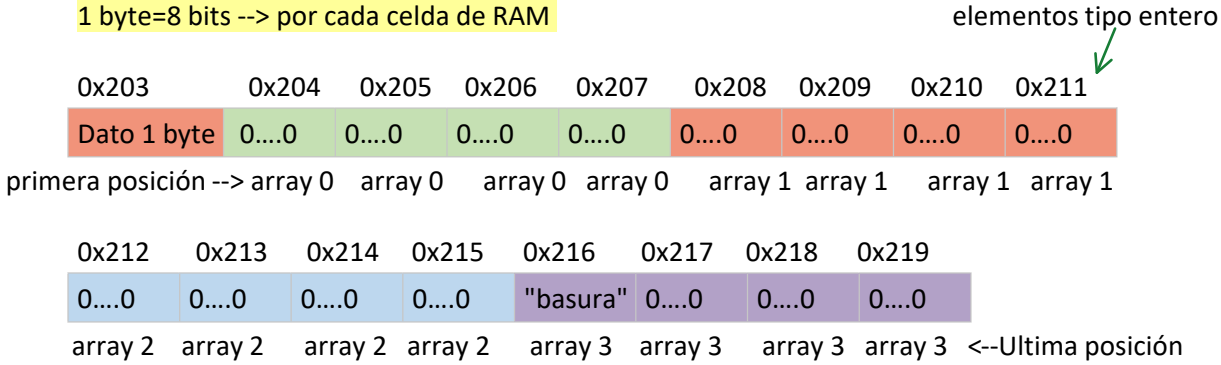
Almacenamiento en memoria para una variable:



0x10	0x11	0x12	0x13
1111 1111	0000 0000	0000 0000	0000 0111
miVar	miVar	miVar	miVar

Almacenamiento en memoria para un arreglo
int array[4]; //declaración de arreglo para 4 elementos.

1 byte=8 bits --> por cada celda de RAM



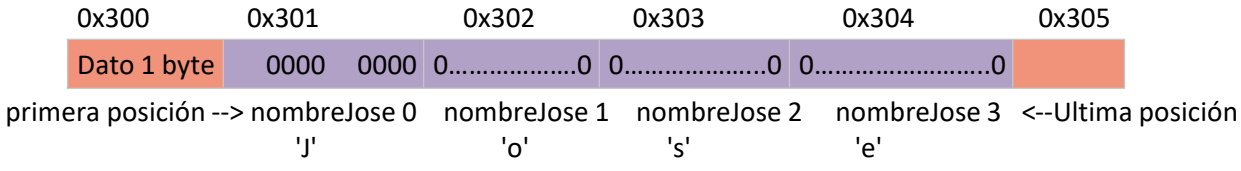
Inicialización de los arreglos

//Declaración e inicialización de un arreglo
<tipo de dato> identificador[<<Tamaño>>]={<valorPos0>, <valorPos1>,..., <valorPosN-1> };

Ejemplo:

```
char nombreJose[4]={'J','o','s','e'};
```

En memoria física se tendría lo siguiente:



```
int numerosEnterosPositivos[3]={0,2,6};
```

Nota: Es importante saber que solamente se pueden colocar todos los valores al mismo tiempo cuando se declara un arreglo.

Esto es un error de sintaxis de inicialización:

```
numerosEnterosPositivos[3]={0,2,6};     numerosEnterosPositivos[]={0,2,6};
```

Sintaxis de acceso a los arreglos, ya sea para lectura o escritura:

```
//Escritura de datos por posición
<<identificador del arreglo>>[posModificar]=<<nuevoValor>>;

//Lectura de datos por posición
<variableExterna>=<identificador del arreglo>[posLectura];
```

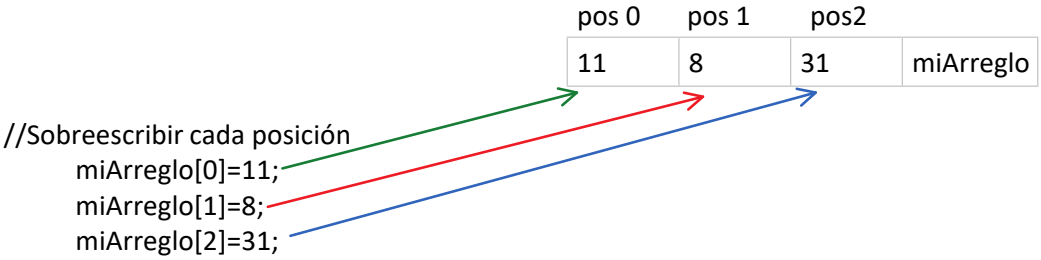
Ejemplos:

//Declaración arreglo entero vacío de 3 elementos

```
int miArreglo[3];
```

pos 0 pos 1 pos2

int miArreglo[3];



//Sobreescribir cada posición
miArreglo[0]=11;
miArreglo[1]=8;
miArreglo[2]=31;

Ejercicio:

Int arr[10]={……}

Ejemplo

martes, 21 de septiembre de 2021 07:17 a. m.

```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

#define TAM 12
//arreglos globales

void main(){
//arreglo local
	//arreglo unidimensional de 12 elementos
	int arreglo[TAM]={1,2,3,4,5,6,7,8,9,10,11,12};

	int indice=0;  //se utiliza i  o j o k como indices para recorrer arreglos.

	for(indice=0; indice<TAM; indice++){
		printf("El elemento en posición %d es %d\n",indice, arreglo[indice]);
	}

	//While
	while(indice<TAM){
		printf("El elemento en posición %d es %d\n",indice, arreglo[indice]);
		indice++;
	}

}
```

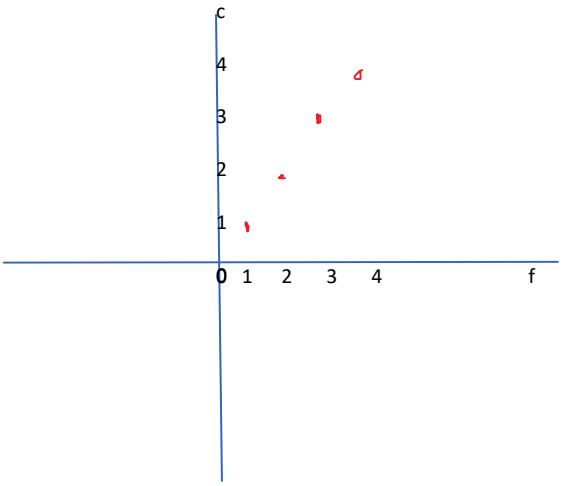
```
#include <stdio.h>
#include <stdbool.h>
#include <math.h>

#define TAM 80000000
//arreglos globales    Sección de memoria estatica y crece conforme lo solicita el programa
int arregloBig[TAM];

void main(){
//arreglo local  en memoria Stack->4000 Kb
	//arreglo unidimensional de 12 elementos
	//int arreglo[TAM]={1,2,3,4,5,6,7,8,9,10,11,12};

	//int arregloCopia[TAM];

	//char arreglo[TAM]={'a','z','x','5','p','3','4','$','#','g','a','!'};

	//float arreglo[TAM]={0.1,10.2, 10.3, 11.4, 0.65, 5.5};

	//Se declara un arreglo vacio para almacenar 1000000
```

```
	int indice=0;  //se utiliza i  o j o k como indices para recorrer arreglos.

	//Se recorre el arreglo original
	/*for(indice=0; indice<TAM; indice++){
		printf("El elemento en posición %d es %d\n",indice, arreglo[indice]);
	}*/

	//Se realiza una copia de datos
	/*for(indice=0; indice<TAM; indice++){
		printf("El elemento original en el arreglo en la posición %d es %d\n",indice,
		arreglo[indice]);
		arregloCopia[indice]=arreglo[indice]%2;  //Se resguarda el valor original con modulo 2
		en nuevo arreglo
	}*/

	//Se realiza un segundo recorrido
	//indice=0; //se reinicia el contador de posiciones del arreglo
	/*
	  while(indice<TAM){
		printf("El elemento en posición %d es %d\n",indice, arregloCopia[indice]);
		indice++;
	}*/

	int i;
	for(i=0;i<TAM;i++){
		printf("El valor actual de i es: %d\n", i);
		arregloBig[i]=i;
	}

	for(i=0;i<TAM;i++){
		printf("El elemento en posición %d es %d\n",indice, arregloBig[i]);
	}

}
```

Arreglos Multidimensionales

martes, 21 de septiembre de 2021 08:11 a. m.

Un arreglo en C puede tener una, dos o más dimensiones. Por ejemplo, un arreglo de dos dimensiones también llamado matriz, es interpretado como un arreglo unidimensional de dimensión "F"(filas), donde cada elemento de "F" es un arreglo unidimensional de dimensión "C" (Columnas). Entonces un arreglo de dos dimensiones, contiene "F"*"C" elementos.



FILAS

Arreglo matricial

0,0 = X	0,1=X3	…	0,m
1,0=X2			
…			
n,0			n,m

COLUMNAS

Sintaxis de declaración de arreglos matriciales en C:

<tipo de dato> identificador[TAM D1][TAM D2];

Sintaxis de declaración de arreglos multidimensionales en C:

<tipo de dato> identificador[TAM D1][TAM D2]…[TAM DN];