La clase pasada, estuvimos en contacto con los comandos de control, las funciones que nos daban un poco de libertad para crear lo que nosotros queramos con respecto a comparaciones. Hoy vamos a ver algo pesado pero muy útil y corresponde a los **arreglos** (**arrays**) y **ordenamiento** de variables.

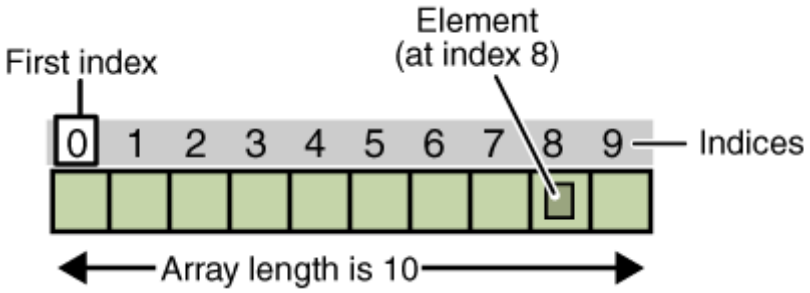
Un arreglo o array es una serie de variables del mismo tipo de manera corrida, bastante usada para crear listas o varios objetos de una misma clase. La manera de usarlo es:

```
TipoDeVariable NombreVariable[x]; //x es la cantidad de objetos en el array
```

Por ejemplo, un array de 5 enteros sería:

```
int miArray[5];
```

Y aquí podemos guardar **5 enteros**. Los lugares **empiezan desde el cero** y el último elemento, luego del array, es el valor “**NULL**” o **Nulo** que indica el fin de la cadena.



Si quisiésemos ocupar el array con los números que van del **1 al 5** se podría hacer así:

```
int main (void)
{
    int miArray[5] = {1,2,3,4,5};

}
```

O así:

```
    int miArray2[5];

    miArray[0] = 1;
    miArray[1] = 2;
    miArray[2] = 3;
    miArray[3] = 4;
    miArray[4] = 5;
```

Vean que el primer número está en el lugar 0 del array, y va hasta el lugar 4, llenando **5 espacios en total**. También que uno puede pasar a través de los valores del array llenando el espacio entre los corchetes con el número referente. Es muy importante, porque si quieren sacar algo de **miArray[5] no verán un valor** y se estarán preguntando qué están haciendo mal como si de magia negra se tratase.

Una forma mas divertida de llenar estos espacios es así:

```c
int main (void)
{
    int miArray[5];
    int i; //contador

    for (i = 0; i<5; i++)
    {
        miArray[i] = i+1;
    }
}
```

Es decir que con esta fórmula **podemos pasar por todos los elementos del array para llenar cada uno de los espacios**, sin tener que crear un código muy extenso. La variable **i** hace de contador para pasar por cada elemento del array (en cada for, **i aumenta**, haciendo que **miArray[i] cambie** en cada uno de los bucles) y además me sirvió para usarlo de relleno de datos, ya que seguía un patrón.

Yyyyyyyyyy acabo de darme cuenta que jamás dije cómo puede, el usuario, **ingresar datos desde el teclado**:D. La función contraria a printf es:

```
scanf("%d", variable);
```

En este caso, %d corresponde al ingreso de una variable entera. Aquí la tabla de referencias:

%c	Entrada de un carácter
%d	Entrada de un número entero
%f	Entrada de un número real
%s	Entrada de una cadena

Entonces, si quisiésemos meter por teclado los 5 valores del array, serían:

```c
#include <stdio.h>

int main (void)
{
    int miArray[5];
    int i; //contador

    for (i = 0; i<5; i++)
    {
        printf("\nInserte un numero entero:");
        scanf("%d",miArray[i])
    }
}
```

Fácil, ¿Cierto? Como queremos introducir un **número entero**, scanf hace referencia a eso y además involucra a la **variable** a rellenar. El printf se usa para hacer mas amigable la entrada del usuario.

Ahora que vimos esto, vayamos al grano con el **ordenamiento**. Supongamos que tenemos un **array** de **5 elementos**, con valores enteros aleatorios y tenemos que ordenar esos elementos de menor a mayor.

Primero que nada, debemos pasar por los 5 elementos para **compararlos**, así que de seguro va un **for**:

```c
#include <stdio.h>

int main (void)
{
    int miArray[5] = {4,2,1,5,3};

    for(i = 0; i < 5; i++)
    {

    }
}
```

Esto lo hice con 5 elementos y con esos valores, pero pueden cambiar ambas variables

Me olvidé de declarar el **entero i**, que es nuestro **contador** en el for.
¿Qué pasaría en caso de que encuentre, en el barrido del array, **un número mas grande que otro**? Debería **cambiarlo de posición**. Entonces, creo una **variable auxiliar** que me ayude a esto.

```c
#include <stdio.h>

int main (void)
{
    int miArray[5] = {4,2,1,5,3};
    int i; //contador
    int aux; //variable auxiliar, para cambiar valores

    for(i = 0; i < 5; i++)
    {
        if(miArray[i] > miArray[i+1])
        {
                    //cambio de valores
                        aux = miArray[i];
                        miArray[i] = miArray[i+1];
                        miArray[i+1] = aux;
        }
    }
}
```

Cada vez que pasa por un valor, lo va a **comparar** con su elemento siguiente (es decir, la primera vuelta comparará 4 con 2). **Si esta condición se cumple**, **entonces intercambian lugares**. Primero guardo uno de los valores en una variable auxiliar, para no perderla. Luego copio un valor en el otro, y por último la variable auxiliar dejará el valor en el otro espacio.

Esto se hará una sola vez, haciendo que quede de alguna manera extraña. Vamos a comprobarlo:

```c
    //imprime los valores
    for(i=0;i<5;i++)
    {
                printf("%d:%d\n",i,miArray[i]);
    }
```

Agrego la función para **imprimir** cada uno de los **valores en pantalla**, y luego **compilo y ejecuto**:

```
C:\Users\w7\Desktop>Untitled1.exe
0:2
1:1
2:4
3:3
4:5
```

El primer número corresponde al contador y el segundo al valor en el array.

En definitiva, este ordenamiento que hicimos no está mal pero **debemos repetirlo varias veces**. **¿Cuántas?** Bueno, la longitud del array menos 1. **¿Por qué?** Porque este proceso se debe repetir **tantas veces como elementos tenga en el array**, para dejarlos todos ordenados. Con la única diferencia que **no es necesario hacerlo con el**

último valor de todos, porque cuando haya pasado 4 veces (en este caso), ya sabemos que el número que viene primero es el más chico de todos, comparado con los otros 4 elementos de la cadena. Como siempre intentaremos de hacer nuestro código más eficiente, **cuando no es necesario no va dentro**. Entonces por bucle de repetición:

```
#include <stdio.h>

int main (void)
{
    int miArray[5] = {4,2,1,5,3};
    int i; //contador
    int j; //contador2
    int aux; //variable auxiliar, para cambiar valores

    for(j = 0; j<4; j++)
    {
      for(i = 0; i < 5; i++)
      {
          if(miArray[i] > miArray[i+1])
          {
                        //cambio de valores
                           aux = miArray[i];
                           miArray[i] = miArray[i+1];
                           miArray[i+1] = aux;
          }
      }
    }

    //imprime los valores
    for(i=0;i<5;i++)
    {
                    printf("%d:%d\n",i,miArray[i]);
```

Agregamos otro for, porque conocemos cuántas veces se debe repetir, la variable j que corresponde a otro contador y por último nos faltará ejecutar el programa:

```
C:\Users\w7\Desktop>Untitled1.exe
0:1
1:2
2:3
3:4
4:5
```

Perfectamente ordenado :).

Intenten practicar haciendo lo mismo pero desde cero, con más elementos y, si se animan, métanse con matrices:).

Saludos futuros hackers.

Pueden seguirme en Twitter: @RoaddHDC

Cualquier cosa pueden mandarme mail a: r0add@hotmail.com

Para donaciones, pueden hacerlo en bitcoin en la dirección siguiente: 1HqpPJbbWJ9H2hAZTmpXnVuoLKkP7RFSvw

Roadd.
