

UNIVERSIDAD DE GUANAJUATO

DIVISIÓN DE INGENIERÍAS CAMPUS IRAPUATO-SALAMANCA

LIC. EN ING. EN SISTEMAS COMPUTACIONALES

ALGORITMOS Y ESTRUCTURAS DE DATOS

PROFESOR: DR. CARLOS HUGO GARCÍA CAPULÍN

NO. DE TAREA: 03

NOMBRE DE LA TAREA:

ARREGLOS (DINÁMICOS)

ESTUDIANTE:

MANRÍQUEZ COBIÁN ROGELIO

FECHA DE ENTREGA:

11 DE SEPTIEMBRE DEL 2020



Problema

Realizar la captura de datos de un arreglo y mostrarlo en pantalla.

Copiarlos datos del arreglo en otro, de tal forma que queden en orden invertido y mostrar el resultado en pantalla.

Por ejemplo:

Arreglo1:

10

20

30

40

50

Arreglo2:

50

40

30

20

10

NOTA: Para la codificación del programa crear los arreglos de forma dinámica y manipular los arreglos utilizando apuntadores.

Solución Implementada:

```

21  */
22
23  #include <stdio.h>
24  #include <stdlib.h>
25
26  int main (void){
27      int nD;
28      int *ptr1;
29      int *ptr2;
30
31      //Escribiendo el arreglo sobre el apuntador
32      printf ("Numero de datos a calcular: ");
33      scanf ("%i", &nD);
34
35      //Declaración de arreglos en forma dinámica
36      //apuntador a un entero hacia ptr1 y ptr2
37      ptr1 = (int *)malloc(nD*sizeof(int)); //malloc asignación de memoria
38      ptr2 = (int *)malloc(nD*sizeof(int));
39
40      //Ver si se asignó correctamente la memoria
41      if (ptr1 == NULL && ptr2 == NULL){
42
43          printf ("Error al asignar memoria\n");
44          exit (0); //salir del programa exit (0); = return 0;
45      }
46
47      //Capturar datos
48      for (int i=0; i<nD; i++)
49      {
50          printf("Dato [%i]: ", i);
51          scanf ("%i",ptr1+i); //&datos[i];
52      }
53
54      //Invertir datos
55      for (int i=0; i<nD; i++)
56      {
57          *(ptr2+i) = *(ptr1+nD-i-1);
58      }
59
60      printf("\n");
61
62      //Asignacion de los datos capturados mediante el puntero ptr1
63      printf("Los datos capturados son:\n");
64      for (int i=0; i<nD; i++)
65      {
66          printf("Dato [%i]: %i\n", i, *(ptr1+i));
67      }
68
69      printf("\n");
70
71      //Mostrar datos invertidos del puntero ptr2
72      for (int i=0; i<nD; i++)
73      {
74          printf("Dato [%i]: %i\n", i, *(ptr2+i));
75      }
76
77      //Eliminar la memoria dinámica
78      free(ptr1);
79      free(ptr2);
80      getchar();
81      return 0;
82
83  }

```

Pruebas y Resultados:

A continuación, se irá describiendo paso por paso lo que hizo en la sección “Solución Implementada” en la cual comenzamos a escribir nuestro código en lenguaje C, con las librerías “**#include <stdio.h>**” la cual ayuda a que el código se muestre en pantalla, y la otra librería “**#include <stdlib.h>**” la cual es una librería estándar que nos ayudará con varias reservadas para la escritura del código.

NOTA: Guardar el archivo como T03.c

Comenzamos escribiendo nuestra función “main” donde escribiremos nuestro algoritmo.

Lo primero que haremos es declarar nuestras variables donde almacenaremos los datos, estas variables serán de tipo “**int**” en donde nuestra primera variable será “**nD**” la cual servirá para pedir cuantos datos quiere el usuario a almacenar, y las otras dos variables serán apuntadores “**ptr1** y **ptr2**” donde el usuario empezará a escribir los datos que quiere almacenar en el primer apuntador.

Ahora solicitamos, cuantos datos el usuario quiere almacenar en “**nD**”;

Después de esto haremos nuestra declaración de los arreglos/apuntadores (**ptr1** y **ptr2**) de la manera dinámica utilizando la memoria dinámica que nos proporciona la librería **stdlib.h** llamada “**malloc**” en el cual le proporcionaremos el tamaño de memoria que el usuario solo desea utilizar.

Para observar que la asignación de memoria se hizo de manera correcta haremos una pequeña condición donde “**ptr1** y **ptr2** son idénticos a **NULL**” haremos una impresión en pantalla donde se indicará “**Error en la asignación de memoria**”.

Ahora lo que haremos es ir capturando cada dato que el usuario desea almacenar con ayuda de un for, en el tendrá como criterio que el índice de nuestro ciclo iniciará en cero, la variable de control será “**i<nD**” y en cada iteración incrementará en uno.

Nuestra condición dentro del ciclo será ir pidiendo cada dato, en nuestro apuntador/arreglo e irlo almacenando ahí, sabiendo que “(**ptr1 + i**) es igual al escribir **&datos[i]**”

Ahora que ya tenemos los datos almacenados en nuestro apuntador, tenemos que invertir los datos de la siguiente manera:

- nD: 3
 - Dato [0]: 3 -> Dato [0]: 1
 - Dato [1]: 2 -> Dato [1]: 2
 - Dato [2]: 1 -> Dato [2]: 3

Es por esto por lo que se implementó un algoritmo para llegar a esta solución en la cual se explicará de la siguiente manera:

Iniciaremos escribiendo un ciclo for con nuestro índice iniciado en cero, la variable de control será “**i<nD**” en cada iteración incrementará en uno.

Dentro de nuestro ciclo tenemos la condición de que copiaremos cada dato que está dentro de “**ptr1 + nD – i – 1**” se lo estará pasando al nuevo arreglo/apuntador de una manera invertida “**ptr2**”.

Después de esto solo daremos a mostrar los datos que se capturaron en el arreglo “**ptr1**” con el mismo ciclo que hemos implementado con anterioridad y haremos que nuestra condición vaya mostrando dato por dato en la dirección almacenada en “**ptr1**”.

De igual manera se realizará un ciclo como se mencionó en el cual ahora imprimiremos en pantalla los datos invertidos en las posiciones que el usuario desea ingresar; todo esto estará establecido en nuestra condición en la que “**ptr2**” almacenó los datos cuando se hizo el intercambio de valores en nuestro ciclo.

Para finalizar nuestro código solamente tenemos que liberar/eliminar la memoria que utilizamos en los apuntadores/arreglos con la línea de código “**free(ptr1), free(ptr2)**”, ya que si no se hace una correcta eliminación de memoria ésta la perdemos por algún lado de nuestra computadora y será muy algo tedioso de eliminarla.

Para comprobar que nuestro programa realiza lo pedido, abriremos nuestro “**CMD**” e ingresaremos hasta la carpeta donde se tiene guardado el archivo; en mi caso se encuentra en la dirección:

- *C:\Users\rmanr\Documents\3_SEMESTRE) 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS*

Y con el comando <dir> veremos que el archivo se ha guardado de manera satisfactoria.

```

cmd.exe

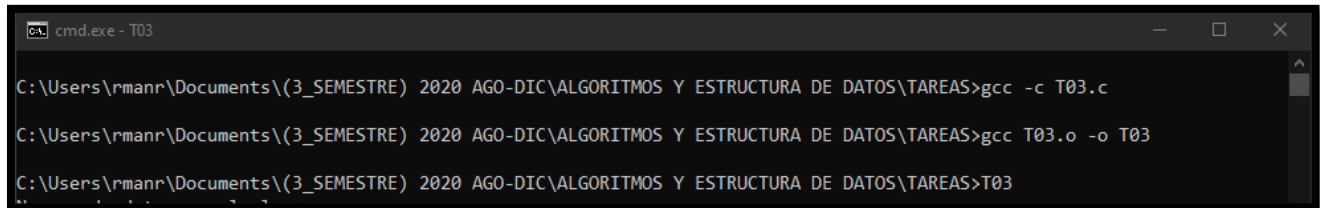
C:\Users\rmanr\Documents\3_SEMESTRE) 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS>cd TAREAS

C:\Users\rmanr\Documents\3_SEMESTRE) 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS>dir
El volumen de la unidad C es OS
El número de serie del volumen es: C6F9-7B98

Directorio de C:\Users\rmanr\Documents\3_SEMESTRE) 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS
08/09/2020  08:45 p. m.    <DIR>          .
08/09/2020  08:45 p. m.    <DIR>          ..
08/09/2020  04:05 p. m.         46,328 run3.exe
10/09/2020  04:00 p. m.         1,694 T03.c
08/09/2020  08:45 p. m.         46,328 T03.exe
08/09/2020  08:45 p. m.         1,738 T03.o
                4 archivos          96,088 bytes
                2 dirs  854,924,963,840 bytes libres

C:\Users\rmanr\Documents\3_SEMESTRE) 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS>
  
```

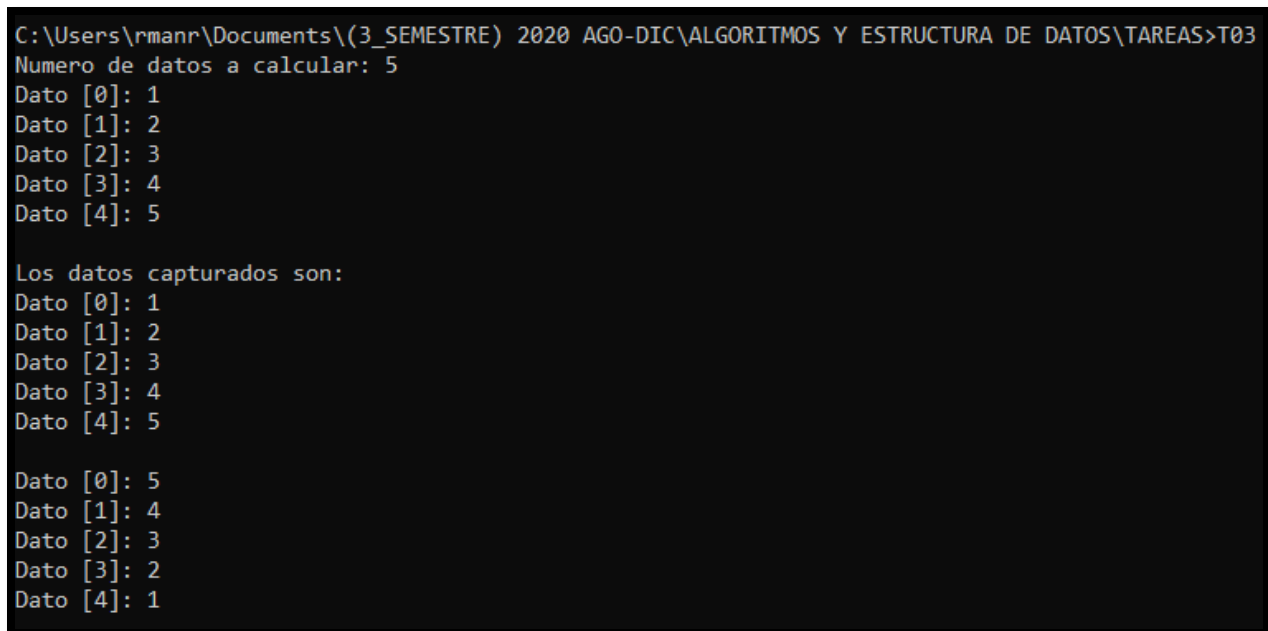
Ahora tendremos que compilar nuestro código con los siguientes comandos que se muestran en la imagen:



```
cmd.exe - T03
C:\Users\rmanr\Documents\3_SEMESTRE 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS>gcc -c T03.c
C:\Users\rmanr\Documents\3_SEMESTRE 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS>gcc T03.o -o T03
C:\Users\rmanr\Documents\3_SEMESTRE 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS>T03
```

Si todo salió bien, solamente dará líneas de salto significando que todo el proceso de compilación y enlazamiento salió bien; de lo contrario, escribiste algún comando mal o tu código tiene algún error de sintaxis.

Ahora ejecutaremos nuestro programa para realizar las pruebas y observar si es lo que se quiso resolver desde un principio



```
C:\Users\rmanr\Documents\3_SEMESTRE 2020 AGO-DIC\ALGORITMOS Y ESTRUCTURA DE DATOS\TAREAS>T03
Numero de datos a calcular: 5
Dato [0]: 1
Dato [1]: 2
Dato [2]: 3
Dato [3]: 4
Dato [4]: 5

Los datos capturados son:
Dato [0]: 1
Dato [1]: 2
Dato [2]: 3
Dato [3]: 4
Dato [4]: 5

Dato [0]: 5
Dato [1]: 4
Dato [2]: 3
Dato [3]: 2
Dato [4]: 1
```

A continuación, se describirá de manera breve los resultados obtenidos por el programa que se realizó:

El programa comienza por pedirnos el número de datos:

Nosotros podemos ingresar cualquier número, pero para el ejemplo se agregó el número 5.

Ahora nos estará pidiendo el dato a guardar en cada arreglo dentro del apuntador; aquí se aplicó un conteo del 1-5.

Después de esto el programa muestra la captura de los datos y después de esto muestra el arreglo con los datos invertidos.

Vemos que el problema se ha solucionado de manera satisfactoria, y cumple con lo pedido.

Con esto nos queda en claro la manera sencilla de cómo es trabajar con apuntadores con arreglos, y es muy interesante la manera en la que se pueden manipular los datos, y en la cual podemos realizar diversos programas con la ayuda de estos apuntadores.