



TECNOLÓGICO
NACIONAL DE MÉXICO

Tecnológico Nacional de México
Instituto Tecnológico de Chihuahua
Departamento Eléctrica – Electrónica
Especialidad Sistemas Embebidos
Arquitectura de Prog. Para Hardware
Laboratorio 1.- Estructura de Datos

Alfredo Chacon Aldama

Jesús Adrián Guerra Delgado 19060742

Fecha de entrega:
09 de septiembre de 2022

Antecedentes

Lenguaje C

El lenguaje C fue desarrollado entre 1969 y 1972 por el empleado de los Laboratorios Bell, Dennis Ritchie como resultado de buscar la forma en que se pudiera desarrollar UNIX de forma portable, en lugar de su forma original en lenguaje ensamblador o lenguaje B, desarrollada por el ya mencionado Dennis Ritchie y Ken Thompson, ya que de ese modo solo podía correr en una PDP-7.

Entre las características y ventajas que tiene el lenguaje C nos encontramos con:

- Portabilidad
- Librerías estándar
- Velocidad
- Cantidad de operadores
- Tipos de datos.

En el año 1972 el lenguaje alcanzo un gran auge, mientras que Ken Thompson desarrollo una gran parte del kernel de Unix en este lenguaje en una computadora PDP-11.

Mas adelante la tecnología C y Unix se distribuía de manera gratuita entre la universidades. En 1978 de lanza la primera edición del libro “The C Programming Language” escrita por Brian Kernighan y Dennis Ritchie.

En 1983 ANSI inicia con la estandarización del Lenguaje C basado en el C original desarrollado por sus creadores. La primera versión de este estándar se completo en 1989 tomando el nombre de dicho año C89, y las versiones siguientes seguirían el mismo patrón.

Conforme fue pasando el tiempo fueron surgiendo diversos lenguajes basados en el lenguaje C. Por ejemplo:

- Objective-C
- C#
- C++
- Perl
- Java
- PHP

Estructuras de datos

Una estructura es una colección de variables con diferentes tipos de datos. Constituyen un medio excelente para agrupar un conjunto de elementos heterogéneos y manejarlos como un todo, facilitando así el manejo de datos complejos.

La combinación de varios tipos de datos especiales para formar estructuras de datos únicas y dependientes de las aplicación, puede simplificar el diseño de nuestro proyecto.

Las estructuras son un mecanismo que permite a un ingeniero de software organizar los datos dentro de un programa:

- Fundamentalmente le permiten juntar y asociar ciertos tipos, y crear un contenedor especial para el conjunto de variables.
- Las estructuras se pueden combinar con muchas características diferentes de nuestros lenguajes de programación para permitirnos realizar operaciones especiales.

Las estructuras de datos se dividen en dos ramas:

- Estructuras de Datos compuestos.- Compuestas por miembros creados con los tipos de C incorporados.
- Estructuras de datos abstractos.- Compuestas por un conjunto de funciones que toman un parámetro de la estructura para luego operar con él.

Arreglos

Desde el punto de vista del programa, una matriz (array o vector) es una zona de almacenamiento contiguo, que contiene una serie de elementos del mismo tipo. Mientras que desde el punto de vista lógico podemos considerarlas como un conjunto de elementos ordenados en fila.

Si bien todas las matrices son de una dimensión, los elementos de esta fila pueden generar a su vez mas matrices, tantas como sean necesarias, pero en general se usan las de dos o 3 dimensiones.

Características de los arreglos:

- Puede contener cualquier número de elementos.
- Los elementos deben ser del mismo tipo.
- El índice está basado en cero.
- El tamaño del arreglo debe especificarse en la declaración.

Para acceder a cualquier elemento de un arreglo es necesario escribir el nombre de dicho arreglo seguido por el número de elemento con que deseamos trabajar entre corchetes, por ejemplo “arreglo_de_ejemplo[4]”. En el caso de que el arreglo sea multidimensional para acceder a sus elementos es similar a como escribimos las coordenadas de un punto en un plano, escribiendo la posición de la primer dimensión, una coma y la otra posición, esto aplicando para un arreglo de dos dimensiones, por ejemplo “arreglo_de_ejemplo[2,3]”.

Apuntadores o punteros

Los Punteros son un tipo especial de datos en C y sus derivados cuyo fin específico es almacenar direcciones de objetos. Comparten características con las variables como son:

- ✓ Tienen un valor
- ✓ Pueden ser asignados
- ✓ Tienen un algebra específica
- ✓ Pueden ser almacenados en matrices
- ✓ Pueden ser pasados y devueltos como parámetros en una función

En general se dice que un puntero “apunta” o “señala” a un objeto determinado cuando su valor es el de la dirección en memoria del objeto.

Para obtener la dirección de un objeto se usa el operador “&” denominado operador de referencia. Mientras que para obtener el referente a partir del puntero se usa el operador “*” llamado operador de “indirección” o de “dereferencia”.

Objetivo

Diseñar un programa en lenguaje C que utilice estructuras de datos, arreglos, apuntadores, funciones, etc. Para que clasifique microcontroladores según su tamaño de arquitectura (32, 16 y bits).

Metodología

Con el propósito de cumplir con las especificaciones de la tarea se tomo como base los códigos de ejemplo brindados por el docente. De estos se tomo en primer lugar la función “cleanBuffIn” la cual nos ayudó a limpiar el buffer previo a la lectura de los datos, enseguida se diseñará una estructura anidada dentro de otra, la anidada guardará datos del tipo char y mientras que la principal usará datos del tipo entero.

Luego de esto se declaro un arreglo de datos conforme a la cantidad de microcontroladores. Luego de esto se conto y separo cada tipo de microcontrolador en un arreglo de la función definida para cada tipo de arquitectura en particular, para finalmente imprimir cada lista de los microcontroladores registrados. Todo esto dentro del ambiente Devc++.

Material

- IDE Embarcadero-Dev-C++-6.3

Desarrollo

Se realizo el código de la presente practica de la siguiente forma, lo primero como antes se menciono fue el tener listos los códigos de ejemplo de la plataforma para buscar en ellos funciones que pudiesen ser útiles en el trabajo a realizar.

De esta manera se realizo el diseño de dos estructuras, una de ellas, llamada *Datos_S*, guardando solo el nombre y fabricante de los microcontroladores, mientras que la segunda, llamada *Micros_S* guarda el tipo de arquitectura y tiene anidada la primer estructura, Como se muestra en la Figura 1.

```
12  /*****  
13  DECLARACION DE ESTRUCTURA  
14  *****/  
15  typedef struct{  
16      char nombre[20];  
17      char fabricante[20];  
18  }Datos_S;  
19  typedef struct{  
20      int nbits;  
21      Datos_S datos;  
22  }Micros_S;
```

Figura 1. Declaración de las estructuras *Datos_S* y *Micros_S*

A continuación se procedió a tomar la función *cleanBuffIn* de los ejemplos vistos en clase, para limpiar el buffer de entrada, además de la creación una función específica para imprimir en pantalla la portada con los datos solicitados llamada *portada* esto se puede ver a continuación en las Figuras 2 y 3.

```
23  /*****  
24  PROTOTIPO DE FUNCION  
25  *****/  
26  void portada(void);  
27  void cleanBuffIn(void);
```

Figura 2. Declaración de las funciones *portada* y *cleanBuffIn*

```

177 /*****FUNCION portada*****/
178 void portada(void){
179     printf("\t Tecnologico Nacional de Mexico\n");
180     printf("\t INSTITUTO TECNOLOGICO DE CHIHUAHUA\n");
181     printf("\tDepartamento Electrica - Electronica\n");
182     printf("\t Especialidad: Sistemas Embebidos\n");
183     printf("\tArquitectura de Prog. Para Hardware\n");
184     printf("\t Docente: Alfredo Chacon Aldama \n");
185     printf("\t Jesus Adrian Guerra Delgado\n");
186     printf("\t 7mo Semestre\n");
187     printf("\t\tLab1 Estructuras\n");
188     printf("\tClasificador de microcontroladores\n");
189 }
190 /*****FUNCION cleanBuffIn*****/
191 void cleanBuffIn(void){ /*FUNCION PARA LIMPIAR EL BUFFER DE ENTRADA EN SUSTITUCION DE fflush(stdin)*/
192     int ch;
193     while ((ch = fgetc(stdin)) != EOF) /* Brinca o descarta todos los caracteres de stdin, */
194     { /* hasta que se encuentra con EOF, llegado al final del buffer.*/
195         if( ch == '\n' ) break; /* o si encuentra un salto de linea */
196     }
197 }

```

Figura 4. Funciones portada y cleanBuffIn

Lo siguiente que se perpetro fue comenzar a trabajar en la función main, se declararon las variables *cantidad*, *cant8=0*, *cant16=0* y *cant32=32* del tipo entero, la primer variable se usó para guardar la cantidad de microcontroladores que el usuario quiera registrar esto para crear una estructura del tipo *Micros_S* llamada *General* como se muestra en la Figura 4, mientras que las otras 3 se usaron para hacer un conteo de cada tipo de arquitectura, esto se explicara más adelante.

```

42 /*Se piden cuantos microcontroladores se registraran*/
43 printf(" Introduzca la cantidad de microcontroladores:");
44 scanf("%d",&cantidad);
45 Micros_S General[cantidad];

```

Figura 3. Creación de la estructura General

Enseguida se creó la función *registro* esta no devuelve ningún valor y solicita una dirección del tipo *Micros_S* además del valor de una variable entera. Dentro de la función se declaró una variable *x=0* la cual se usó para la búsqueda de errores en el registro del tipo de arquitectura, enseguida se definió la variable *nbits* de la estructura a la que apunta la dirección sea igual a cero. Posteriormente se comenzó a solicitar el nombre y fabricante del microcontrolador, solicitando un dato a la vez a través de la pantalla para luego leerlos con ayuda de un *scanf*, haciendo uso la función *cleanBuffIn* antes de cada *scanf* previniendo la entrada de basura, por su parte el entero solicitado en la llamada a la función *registro* es usado para indicar al usuario el numero de microcontrolador que esta registrando como se puede ver en la Figura 5.

```

151 void registro(Micros_S *micros,int i){
152     /*Declaracion de variables*/
153     int x=0;
154     micros->nbits=0;
155     /*Se leen los datos*/
156     printf("\n Introduzca el nombre del microcontrolador %d:",i+1);
157     cleanBuffIn();
158     scanf("%s",micros->datos.nombre);
159     printf("\n Introduzca el nombre de su fabricante:");
160     cleanBuffIn();
161     scanf("%s",micros->datos.fabricante);
162     printf("\n Introduzca el tipo de arquitectura(8, 16 o 32):");

```

Figura 5. Solicitud de datos dentro de la función registro

Luego se procedió a solicitar el tipo de arquitectura del microcontrolador, para este registro se realizó un ciclo que se repite si es que se presenta algún error, ya sea la entrada de un valor que no sea un numero entero o que el valor no coincida con alguna de las tres arquitecturas con que se esta trabajando, en cada situación el programa muestra un mensaje de error según corresponda y solicita de nueva cuenta el tipo de arquitectura, como se puede ver a continuación en la Figura 6.

```

163      /*Se verifica que se introduzca un dato valido en el tipo de arquitectura*/
164      /*ademas de que sea una de las tres arquitecturas aceptadas*/
165      while(x==0 || (micros->nbits!=8 && micros->nbits!=16 && micros->nbits!=32)){
166          cleanBuffIn();
167          x=scanf("%d",&micros->nbits);
168          if(x==0){
169              printf("\n Error, la entrada es incorrecta");
170              printf("\n Introduzca el tipo de arquitectura otra vez(8, 16 o 32):");
171          }
172          if(x==1 && (micros->nbits!=8 && micros->nbits!=16 && micros->nbits!=32)){
173              printf("\n Error, tipo de arquitectura no soportada");
174              printf("\n Introduzca el tipo de arquitectura otra vez(8, 16 o 32):");
175          }
176      }
177  }

```

Figura 6. Registro del tipo de arquitectura del microprocesador

Como se puede ver en la Figura 7 la función recién expuesta se utilizó dentro de un ciclo *for* en el cual los ciclos a realizar están definidos por la cantidad de microcontroladores a registrar, dentro de este ciclo es que vemos a la función *registro* la cual recibe como datos la dirección de todas las posiciones del arreglo *General* esto haciéndose una por ciclo, además del numero de ciclo que se está realizando.

```

47      /*Funcion de registro de datos*/
48      for(i=0;i<cantidad;i++){
49          registro(&General[i],i);
50      }

```

Figura 7. Empleo de la función registro dentro de main

A continuación se creó la función *checador* esta función como su nombre indica ayudo a revisar si el tipo de arquitectura de un microcontrolador coincide con el que se está buscando, si esto es así, como se puede ver en la Figura 8, regresa un uno de lo contrario envía un cero. Ahora bien como datos de entrada, al hacer un llamado a la función, recibe una dirección del tipo *Micros_S* y la

```

141      /******FUNCION checadore******/
142      int checadore(Micros_S *micros,int arquie){
143          /*Declaracion de variables*/
144          int si_es=0;
145          if(micros->nbits==arquie){
146              si_es=1;
147          }
148          return si_es;
149      }

```

Figura 8. Función checadore

guarda en un apuntador llamado *micros* y de igual manera recibe un entero que almacena con el nombre de *arqui* el cual nos dará el tipo de arquitectura que se quiere buscar.

La función *checador* se utilizó en diversas ocasiones dentro de *main*, en primer lugar se utilizó para contar cuantos microcontroladores hay de cada tipo de arquitectura, esto con ayuda de un *for*, dentro del cual se dice que *x* guardara el valor que devuelve la función *checador*, en caso de ser uno aumentara en uno la cantidad de microcontroladores de ese tipo. Como se puede apreciar en la Figura 9 los datos entregados en la llamada de la función son la dirección de cada miembro del arreglo *General* además del tipo de arquitectura a contar. Finalizado cada conteo se creó un arreglo de estructuras siendo estos *bists8*, *bists16* y *bists32*.

```
52      /*Se realiza un conteo de cuantos microcontroladores de cada tipo hay*/
53      for(i=0;i<cantidad;i++){
54          x=checador(&General[i],8);
55          if(x==1){
56              cant8++;
57          }
58      }
59      Micros_S bists8[cant8]; /*Se crea un arreglo de estructuras para Los*/
60                               /*microcontroladores de 8 bits*/
61      for(i=0;i<cantidad;i++){
62          x=checador(&General[i],16);
63          if(x==1){
64              cant16++;
65          }
66      }
67      Micros_S bists16[cant16];/*Se crea un arreglo de estructuras para Los*/
68                               /*microcontroladores de 16 bits*/
69
70      for(i=0;i<cantidad;i++){
71          x=checador(&General[i],32);
72          if(x==1){
73              cant32++;
74          }
75      }
76      Micros_S bists32[cant32];/*Se crea un arreglo de estructuras para Los*/
77                               /*microcontroladores de 32 bits*/
```

Figura 9. Conteo de microcontroladores

Enseguida se utilizó la función *checador* para clasificar los microcontroladores en sus respectivos arreglos, similar al ciclo anterior se trata de un *for* y de igual manera se dice que *x* será igual al valor devuelto por la función *checador* y en caso de ser uno se comienza la asignación de un arreglo a otro, como se puede ver en la Figura 10 se ha comenzado con un ciclo *for* de 20 vueltas en el cual se mandan los datos de nombre y fabricante del arreglo *General* al arreglo de clasificación

según sea la arquitectura, para luego pasar de igual manera el tipo de arquitectura correspondiente a ese nombre y fabricante.

```

79      /*Se clasifican los datos de acuerdo al tipo de arquitectura*/
80      for(i=0;i<cantidad;i++){
81          x=checador(&General[i],8);
82          if(x==1){
83              for(k=0;k<20;k++){
84                  bits8[j].datos.nombre[k]=General[i].datos.nombre[k];
85                  bits8[j].datos.fabricante[k]=General[i].datos.fabricante[k];
86              }
87              bits8[j].nbits=General[i].nbits;
88              j++;
89          }
90      }
91      j=0;
92
93      for(i=0;i<cantidad;i++){
94          x=checador(&General[i],16);
95          if(x==1){
96              for(k=0;k<20;k++){
97                  bits16[j].datos.nombre[k]=General[i].datos.nombre[k];
98                  bits16[j].datos.fabricante[k]=General[i].datos.fabricante[k];
99              }
100             bits16[j].nbits=General[i].nbits;
101             j++;
102         }
103     }
104     j=0;
105
106     for(i=0;i<cantidad;i++){
107         x=checador(&General[i],32);
108         if(x==1){
109             for(k=0;k<20;k++){
110                 bits32[j].datos.nombre[k]=General[i].datos.nombre[k];
111                 bits32[j].datos.fabricante[k]=General[i].datos.fabricante[k];
112             }
113             bits32[j].nbits=General[i].nbits;
114             j++;
115         }
116     }
117     j=0;

```

Figura 10. Clasificación de los microcontroladores

La última función creada para este código fue *printclas* la cual recibe una dirección del tipo *Micros_S* y un entero que nos ayudó para indicar que numero de microcontrolador es. Dicha función imprime en pantalla el numero de microcontrolador seguido por el nombre y el fabricante, como se puede ver en la Figura 11.

```

137      /******FUNCION printclas******/
138      void printclas(Micros_S *micros,int i){
139          printf("\n %d.- %s fabricado por %s",i+1,micros->datos.nombre,micros->datos.fabricante);
140      }

```

Figura 11. Función *printclas*

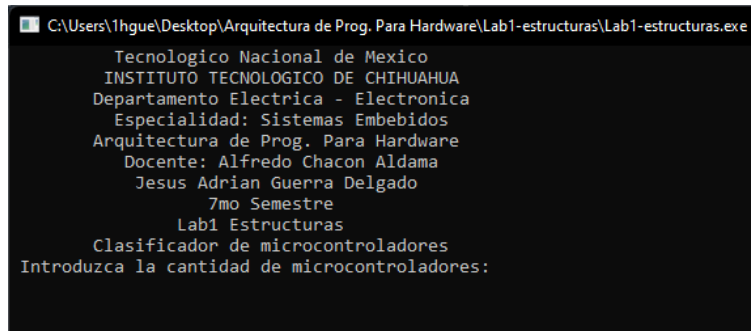
Enseguida se muestra en pantalla un listado de cuantos microcontroladores hay de cada tipo de arquitectura. La función *printclas* como se puede ver en la Figura 12, se implementó en *main* dentro de un ciclo *for* por cada arquitectura, cada ciclo esta definido por el numero de microcontroladores de ese tipo, ya dentro del ciclo se llamó a la función enviándole la dirección de cada miembro de cada uno de los arreglos además del numero de ciclo para ayudar a identificar el numero de microcontrolador que es, dando como resultado una lista de cada tipo de microcontroladores.

```
118      /*Se imprime un inventario de Los microcontroladores registrados*/
119      printf("\n Se tienen %d microcontroladores de 8 bits",cant8);
120      printf("\n Se tienen %d microcontroladores de 16 bits",cant16);
121      printf("\n Se tienen %d microcontroladores de 32 bits",cant32);
122
123      /*Se imprime una lista detallada de cada arquitectura*/
124      printf("\n Los Microcontroladores de 8 bits son:");
125      for(i=0;i<cant8;i++){
126          printclas(&bits8[i],i);
127      }
128      printf("\n Los Microcontroladores de 16 bits son:");
129      for(i=0;i<cant16;i++){
130          printclas(&bits16[i],i);
131      }
132      printf("\n Los Microcontroladores de 32 bits son:");
133      for(i=0;i<cant32;i++){
134          printclas(&bits32[i],i);
135      }
136  }
```

Figura 12. Entrega de la clasificación en pantalla

Resultados

Al momento de ejecutar el código se nos recibe con la portada y nos solicita la cantidad de microcontroladores como se puede ver en la Figura 13.

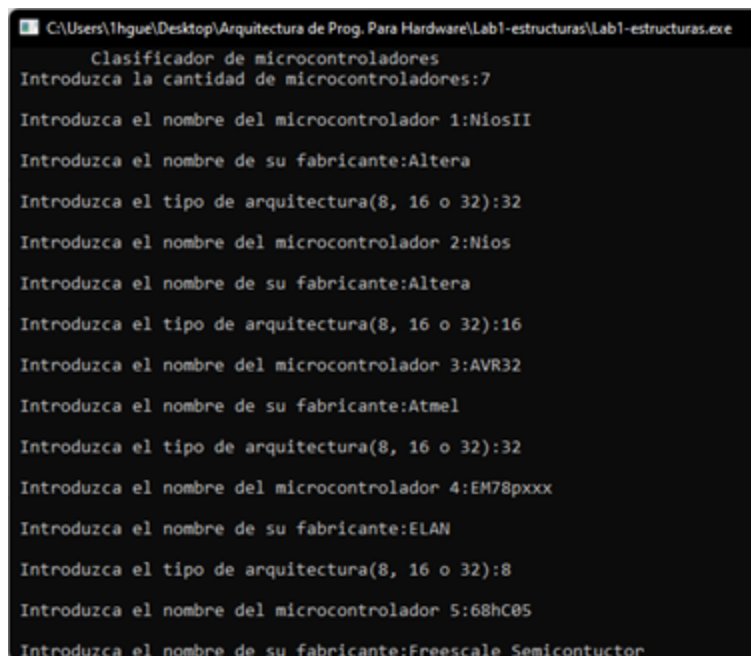


```
C:\Users\Thgue\Desktop\Arquitectura de Prog. Para Hardware\Lab1-estructuras\Lab1-estructuras.exe

Tecnologico Nacional de Mexico
INSTITUTO TECNOLOGICO DE CHIHUAHUA
Departamento Electrica - Electronica
Especialidad: Sistemas Embebidos
Arquitectura de Prog. Para Hardware
Docente: Alfredo Chacon Aldama
Jesus Adrian Guerra Delgado
7mo Semestre
Lab1 Estructuras
Clasificador de microcontroladores
Introduzca la cantidad de microcontroladores:
```

Figura 13. Portada del Programa

Luego de definir cuantos microcontroladores se van a registrar nos pregunta el nombre, fabricante y arquitectura de cada microcontrolador como se puede ver en la Figura 14.



```
C:\Users\Thgue\Desktop\Arquitectura de Prog. Para Hardware\Lab1-estructuras\Lab1-estructuras.exe

Clasificador de microcontroladores
Introduzca la cantidad de microcontroladores:7

Introduzca el nombre del microcontrolador 1:NiosII
Introduzca el nombre de su fabricante:Altera
Introduzca el tipo de arquitectura(8, 16 o 32):32
Introduzca el nombre del microcontrolador 2:Nios
Introduzca el nombre de su fabricante:Altera
Introduzca el tipo de arquitectura(8, 16 o 32):16
Introduzca el nombre del microcontrolador 3:AVR32
Introduzca el nombre de su fabricante:Atmel
Introduzca el tipo de arquitectura(8, 16 o 32):32
Introduzca el nombre del microcontrolador 4:EM78pxxx
Introduzca el nombre de su fabricante:ELAN
Introduzca el tipo de arquitectura(8, 16 o 32):8
Introduzca el nombre del microcontrolador 5:68hC05
Introduzca el nombre de su fabricante:Freescalse Semicontuctor
```

Figura 14. Registro de datos

Finalmente obtenemos cuantos microcontroladores tenemos de cada tipo y una lista de cada una de las tres clasificaciones como se ve en la Figura 15.

```
C:\Users\Ithgu\Desktop\Arquitectura de Prog. Para Hardware\Lab1-estructuras\Lab1-estructuras.exe

Introduzca el tipo de arquitectura(8, 16 o 32):8
Introduzca el nombre del microcontrolador 6:MCS-48
Introduzca el nombre de su fabricante:Intel
Introduzca el tipo de arquitectura(8, 16 o 32):8
Introduzca el nombre del microcontrolador 7:TLC5-870
Introduzca el nombre de su fabricante:Toshiba
Introduzca el tipo de arquitectura(8, 16 o 32):8

Se tienen 4 microcontroladores de 8 bits
Se tienen 1 microcontroladores de 16 bits
Se tienen 2 microcontroladores de 32 bits
Los Microcontroladores de 8 bits son:
1.- EM78pxxx fabricado por ELAN
2.- 68hC05 fabricado por Freescale_Semicontuc
3.- MCS-48 fabricado por Intel
4.- TLC5-870 fabricado por Toshiba
Los Microcontroladores de 16 bits son:
1.- Nios fabricado por Altera
Los Microcontroladores de 32 bits son:
1.- NiosII fabricado por Altera
2.- AVR32 fabricado por Atmel
-----
Process exited after 335.4 seconds with return value 0
```

Figura 15. Resultado de la clasificación de los microcontroladores

Conclusiones

Si bien se crearon 5 funciones para ayudar a la organización del programa, hay una parte del código que también sería conveniente buscar la forma de crear una función para ello, me refiero a la parte de la clasificación de los microcontroladores, sin embargo el código está organizado y bien documentado para ayudar a su entendimiento, parte de lo más difícil fue el detector de errores en la entrada del tipo de arquitectura, mientras que el resto, siendo más específico, el uso de las estructuras y los apuntadores estuvo un tanto más claro gracias a todo lo visto en clase durante las primeras 3 semanas de clase, mientras que más complicado fue un tanto de razonar un poco la manera en que se deben de pasar una cadena de caracteres de un arreglo a otro. De esta práctica destacó la utilidad de las estructuras y apuntadores permitiéndonos crear grupos de variables completamente diferentes y poder llamarlos desde cualquier parte del código sin necesidad de ser redundante o desperdiciar espacio de memoria.

Referencias

Tecnológico Nacional de México. (09 de septiembre del 2022). Material de Clase: “Historia del Lenguaje”. <https://online.fliphtml5.com/qwnxj/mrhn/>.

Tecnológico Nacional de México. (09 de septiembre del 2022). Material de Clase: “U1- Programación Orientada a objetos PUNTEROS”. <https://online.fliphtml5.com/qwnxj/beks/>.

Tecnológico Nacional de México. (09 de septiembre del 2022). Material de Clase: “Arreglos (Matrices) y punteros”. <https://online.anyflip.com/wndov/dvfe/mobile/>.

Tecnológico Nacional de México. (09 de septiembre del 2022). Material de Clase: “Enumeraciones y Estructuras”. <https://www.flipsnack.com/alfredoc/estructuras.html>.