

Universidad Nacional Autónoma de México

Facultad de Ingeniería

Estructura de Datos y Algoritmos I

Actividad 1: Repaso de lo que aprendí en la asignatura de Fundamentos de Programación.

Torres Oropeza Diego Alberto

24/02/2021



# Parte I. Lenguajes de Programación


Al inicio del curso recuerdo que nuestra profesora primeramente nos enseñó los lenguajes de programación existentes y la resolución de problemas, no recuerdo el orden exacto, pero veremos primero los lenguajes de programación.

En este tema la profesora dividió el grupo en equipos de 5 o 6 personas y a cada equipo le asignó un lenguaje de programación, del que teníamos que realizar una infografía y posteriormente debíamos exponer sobre el lenguaje.

Para hacer la infografía tuvimos subtemas que debíamos cubrir, el creador del lenguaje, la fecha en la que se realizó y la fecha de publicación, lugar de publicación, objetivos por los que se creó y las ventajas y desventajas que tenía dicho lenguaje de programación.

Los lenguajes de programación que mi grupo investigó fueron los siguientes (adjunto imágenes de la infografía realizada por mi equipo):

1. Lenguaje C.
2. C++.
3. Java.
4. Python.
5. Fortran.
6. Ruby.
7. C#.
8. Basic.
9. Logo
10. Cobol.
11. Pascal:



# PASCAL

LENGUAJE DE PROGRAMACIÓN

## ¿QUÉ ES?


Pascal es un lenguaje de programación creado como un lenguaje compacto, eficiente y destinado a alentar las buenas prácticas usando programación estructurada.

## OBJETIVO PRINCIPAL

Es un lenguaje de programación de alto nivel diseñado con el objetivo de enseñar programación estructurada y promover un método disciplinado y elegante de programar.

Realizado por:  
Grupo 10 Fundamentos de Programación  
Torres Oropeza Diego Alberto  
González Ramírez Alberto Javier  
Espinoza Escalona Jocelyn Yanet  
González Próspero Amir Hasam  
Perez Castillo Omar  
Perez Avin Paola Celina de Jesus

**CREADO POR**  
El profesor suizo Niklaus Wirth



**CREADO ENTRE**  
**1968 y 1969**

**PUBLICADO EN**  
**1970**

## VENTAJAS

1. Es un lenguaje que casi se considera pseudo-código, así que puedes ofrecer explicaciones comprensibles (y completas) directamente con código.
2. Que se pueda entender cuando se lee el código.
3. Aquí hablamos de la velocidad de ejecución y el uso eficaz de los recursos del ordenador (sobre todo la memoria).

## DESVENTAJAS

1. Hay pocas funciones (especialmente gráficas) para trabajar. Casi todo lo debes programar.
2. No hay "break" ni "return".

**Bibliografía:**  
Lenguaje de programación Pascal (actualizado a 2010) . (2010, 25 septiembre). Lenguajes de programación. <https://lenguajesdeprogramacion.net/pascal-y-delphi/>  
EnUed. (s. f.). Pascal - EnUed. Recuperado 2 de octubre de 2020, de <https://www.ecured.cu/Pascal>  
Pascal - Overview. (s. f.). Tutorialspoint. Recuperado 2 de octubre de 2020, de [https://www.tutorialspoint.com/pascal/pascal\\_overview.htm](https://www.tutorialspoint.com/pascal/pascal_overview.htm)

## Parte II. Resolución de problemas

En este tema la profesora nos planteaba problemas en enunciados, con dichos enunciados debíamos encontrar los datos de entrada y en base a ello encontrar los de salida, pasando por un proceso de la operación o algoritmo que se debía hacer para obtener los datos de salida y también cómo se le mostrarían las cosas al usuario (el pedir los datos y enseñar los resultados).

A continuación le presento un ejemplo de los ejercicios que realicé en este tema:

Una tienda de mayoreo ha puesto en oferta a la venta cierto producto, ofreciendo un descuento del 25% por la compra de más de 3 docenas y 15% en caso contrario. Además, por la compra de más de 3 docenas se obsequia una unidad del producto por cada 3 docenas. Analice el problema y determine el monto de la compra, el monto del descuento, el monto a pagar y el número de unidades de obsequio por la compra de cierta cantidad de docenas del producto. Costo docena \$100

### 1. Inicio

Tienda de mayoreo

### 2. Análisis de requerimientos

- Número de docenas a comprar.
- 1 docena = \$100
- Aplicar descuentos según el # de docenas
- Regalar 1 artículo según el # de docenas
- Precio a pagar con descuentos aplicados según el # de docenas.
- # de regalos a dar según el # de docenas.

### 3. Restricciones

- No se puede comprar menos de una docena.
- No se regala el artículo con Docenas  $\leq 3$ .
- No se aplica el descuento de 25% con Docenas  $\leq 3$ .
- Se aplica el descuento de 15% con Docenas  $\geq 1$ .

### 4. Datos de entrada

- # de docenas (valores positivos mayores de 0)

### 5. Proceso

1.- INICIO

2.-Imprimir "Dame el número de D a comprar"

3.- Leer D.  $D \geq 1$

Si  $D > 3$

Precio final =  $(\#D)(100)(.25)$

SiNo  $D \leq 3$

Precio final=  $(\#D)(100)(.15)$

Si  $D > 3$

Cantidad de regalos=  $\#D/3$

#### 6. Datos de salida

- Costo a pagar.
- Costo con descuentos aplicados de acuerdo al # de docenas.
- # de regalos según el # de docenas compradas.
- Monto de descuentos.

Este ejercicio no me quedó súper bien porque no tenía ni idea de qué hacer, pero la maestra me dijo que mi idea iba medio bien.

### Parte III. Pseudocódigo

Luego de comenzar a ver la resolución de problemas a mano, escritos y medio mal hechos, comenzamos a ver el pseudocódigo, que nos permite realizar un algoritmo de manera informal, sin la necesidad de checar que esté bien escrito con la sintaxis de algún lenguaje de programación.

Para comenzar a realizar nuestros pseudocódigos utilizamos Pseint, que es un software educativo para los que principiantes en programación, en dicho software realizamos diferentes ejercicios, cada uno de ellos requería un algoritmo que debíamos crear para lograr obtener lo que el enunciado pedía.

#### Parte III.I. Bases del pseudocódigo

Empezamos viendo cómo mostrar textos en pantalla y cómo leer datos que el usuario introducía, además, nos dijo la maestra que siempre debíamos poner “;” al final de cada línea de código, después seguíamos avanzando viendo cada vez más funciones, a continuación están las funciones que más usamos en Pseint:

- Escribir: Nos permite mostrar texto en pantalla. Ejemplo: Escribir “Hola”;
- Leer: Nos permite guardar X dato que escriba el usuario en una variable predefinida. Ejemplo: Definir n Como Entero; Escribir “Dame un número”; Leer n;
- Definir: Nos sirve para definir variables, las variables pueden ser definidas en números enteros o real, en lógicos o como un carácter. Ejemplo: Definir x Como Carácter;
- Asignar: Nos sirve para asignarle un valor a una variable. Ejemplo:  $n \leftarrow n+1$ ;

Los 4 que acabamos de ver son muy simples y son prácticamente la base del pseudocódigo y de los demás lenguajes de programación, solo que lo que cambia es la sintaxis de las instrucciones obviamente.

#### Parte III.II. Estructuras condicionales

Luego de ver las instrucciones más básicas de pseudocódigo, comenzamos a ver estructuras condicionales, por ejemplo, el “Si”, que tiene la siguiente estructura:

Si \_\_\_\_ Entonces

SiNo

FinSi

Esta estructura nos permite hacer una acción o una serie de acciones siempre y cuando se cumpla X condición.

Después vimos el “Según”, que nos permite realizar diferentes acciones con el valor de una variable, por ejemplo, tengo la variable “numero”, esta variable puede almacenar números del 1 al 3, entonces tenemos la siguiente estructura:

Segun numero Hacer

1:

2:

3:

FinSegun

Como podemos ver, según el valor de “numero”, se realizarán las instrucciones que el número introducido contenga. Dentro de esta estructura podemos poner un “Si” al inicio, en el cual la condición sea que número necesariamente se encuentre entre 1 y 3, si es así, que se realice el poder escoger la opción, y si no es así, que el algoritmo marque un error.

### Parte III.III. Estructuras cíclicas

Ya vistos y repasados los temas básicos, más los condicionales, comenzamos a ver las estructuras cíclicas, en las que vimos el “Para”, el “Mientras” y el “Repetir”.

Las 3 instrucciones sirven para lo mismo, en base a X condición se repite una serie de acciones, pero la diferencia que tienen estas 3 es su estructura, por ejemplo:

“Para”: En ella asignamos un valor inicial a una variable, y hasta un valor final, con paso designado, se hace una secuencia de acciones:

Para variable\_numerica<-valor\_inicial Hasta valor\_final Con Paso paso Hacer

secuencia\_de\_acciones

“Mientras”: Aquí establecemos que mientras la variable tenga X valor, hará las acciones:

Mientras expresion\_logica Hacer

secuencia\_de\_acciones

FinMientras

“Repetir”: Aquí decimos que se repita X acción(es) hasta que una variable llegue a un valor:

Repetir

secuencia\_de\_acciones

Hasta Que expresion\_logica

Luego de haber repasado todas estas funciones y haber visto algunas otras, realizamos diversos ejercicios e incluso un examen, en el que no me fue muy bien por cierto T\_T.

## Parte IV. Programando en Lenguaje C

Una vez terminado de familiarizarse a la programación en Pseint, comenzamos a programar en Lenguaje C, que, como ya vimos anteriormente, es un lenguaje de programación.

Dentro del Lenguaje C vimos las instrucciones que se ocupan básicamente, como lo que dije en Pseint, las instrucciones con diferente sintaxis.

Dentro del Lenguaje C hay librerías, cada librería contiene instrucciones que nos permiten hacer diferentes cosas cada una, para usar una instrucción X, debemos primero, al inicio del programa, llamar a su librería, de lo contrario, la función no serviría.

### Parte IV.I. Conociendo el Lenguaje C

Las funciones básicas que nos enseñó la profesora fueron las que mostraré a continuación, pero también nos mencionó algunas cosas que debíamos hacer, como siempre poner las librerías que vayamos a ocupar y además, al iniciar a codificar, se pone un `int main(){}`, dentro de las llaves va todo el algoritmo, y obviamente, muy importante poner “;” al final de las instrucciones. Otra cosa que nos comentó es que, al correr un código, la ventana que se abre, en la cual está corriendo nuestro código, se llama ejecutable.

### Parte IV.II. Tipos de datos

En el Lenguaje C existen diferentes tipos de datos que se representan como instrucciones para definir variables con ese tipo de dato:

- `int`: Número entero.(%i)
- `float`: Número con decimales.(%f)
- `double`: Número con pocos decimales.(%d)
- `char`: Carácter.(%c)

Al declarar una variable con alguno de estos tipos de datos, si le queremos dar un valor con `scanf`, ese valor debe ser del mismo tipo de dato, ya que cada tipo de dato tiene una diferente cantidad de almacenamiento, además de que se declaras una variable como `int` es obvio que no le vas a dar valor de 4.

### Parte IV.III. Funciones básicas

Unas funciones básicas, y muy importantes, que nos enseñó la profesora son:

- `printf`: Nos sirve para mostrar texto en el ejecutable:  
`printf("Hola mundo");`
- `scanf`: Sirve como el “leer” en Pseint, para guardar un valor dado por el usuario en una variable. En `scanf` es muy importante especificar qué tipo de dato guardará la variable, a continuación presento la estructura del `scanf` porque tiene unas cosas importantes por explicar:

```
scanf("%i",&n);
```

El “%i” debe ir entre comillas, eso nos denota el tipo de dato a guardar, en este caso, un entero. Enseguida del entrecomillado va una coma y luego un “&” seguido de la variable, el “&” nos indica que en esa variable se va a guardar lo que el usuario escriba, si no ponemos el “&”, no se guardará el contenido en la variable y por ende nuestro código no podría trabajar.

## Parte IV.IV. Cadenas de caracteres y código ASCII

Las cadenas de caracteres son palabras en sí, para poder utilizar palabras como contenido de variables necesitamos la librería <strings.h>, que contiene las instrucciones para guardar cadenas en variables, copiar valor de una variable cadena a otra, comparar cadenas, mostrar cadenas en el ejecutable, etc.

El código ASCII es un código estándar para el intercambio de información, en él todo son caracteres, todos los caracteres en sí, los números y los signos, pero cada uno tiene un número, si queremos mostrar, por ejemplo, una vocal acentuada en el ejecutable con un printf, debemos poner en el lugar de la letra un “%c”, porque la letra acentuada es un carácter como ya dije, luego de esto se debe poner qué letra se quiere en base al código ASCII, ósea, su número.

## Parte IV.V. Condicionales en Lenguaje C

Como en Pseint, en Lenguaje C hay estructura condicionales, esta vez es el “if” y para opciones el “switch”.

- “if”: Es la estructura de “Si” en C, tiene los siguientes componentes:

```
if (n==0)
{
}
else
```

Dentro de los corchetes puedes poner una serie de acciones si se cumple la condición o condiciones que declares en el paréntesis de enseguida del “if”, si no se cumplen las condiciones y quieres dar otra serie de acciones, se pone el “else” que sería como el “SiNo” de Pseint.

- “switch”: Estructura de selección en C, tiene la siguiente forma:

```
switch(opc)
1:
break;
2:
break;
default:
```

Lo de dentro del paréntesis es la variable de la cual dependerá la opción, entre cada número y cada break puedes poner una serie de instrucciones, el break sirve para indicar que hasta ahí llega la acción, si es algo muy grande lo que se hace, se pueden usar las llaves para no tener muchas complicaciones o funciones que veremos más adelante. El “default” actúa como un “SiNo”, sólo que no es exacto obviamente, por ejemplo, tienes opciones del 1 al 5 y el usuario mete un 6, el default sirve para que en casos como este, en el que el usuario introduce datos no permitidos, el código le puede notificar que está mal, es cuestión de decidir si se quiere



presentar en el ejecutable un printf("Opcion no permitida");, o si simplemente quieres dejar que el programa se cierre.

## Parte IV.VI. Estructuras cíclicas en Lenguaje C

Las estructuras cíclicas que vimos fueron 3: "while", "for" y "do-while", las 3 con el mismo propósito de repetir una serie de acciones mientras su condición se cumpla, pero con diferentes formas cada una.

- "for": Se declara un valor inicial del contador, el límite del valor del contador y el incremento o decremento según sea el caso, la estructura es la siguiente:

```
for(i=0;i<=10;i++)  
{  
}
```

Dentro de las llaves van las instrucciones a hacer, "i" es el contador, se debe declarar a inicios del programa, los "++" indican que después de que el ciclo realice una vuelta, se le incrementará 1 a "i", así hasta que "i" llegue a 10 en este caso, aunque el límite no solo puede ser un número, si no también una variable.

- "while": Aquí sólo se declara el límite del contador y dentro del ciclo se le debe dar el incremento:

```
i=0;  
while(i<=10)  
{  
    i++;  
}
```

- "do-while": Es parecido al "while", en el nombre lo dice, pero en este primero se hace la secuencia de acciones y se suma el contador y finalmente comprueba que no se pase del límite:

```
i=0;  
do  
{  
    i++;  
}  
while(i<=10);
```

## Parte IV.VII. Arreglos

La maestra nos enseñó los arreglos después de lo anterior, eran una especie de vectores que iban almacenando datos en cada posición que tenían, un arreglo de 5 podía guardar 5 números, siempre empezaban en la posición 0 y no en la 1, además de números podemos almacenar caracteres o incluso palabras, pero eso ya está más complicado. Para declarar un arreglo utilizábamos algo así:

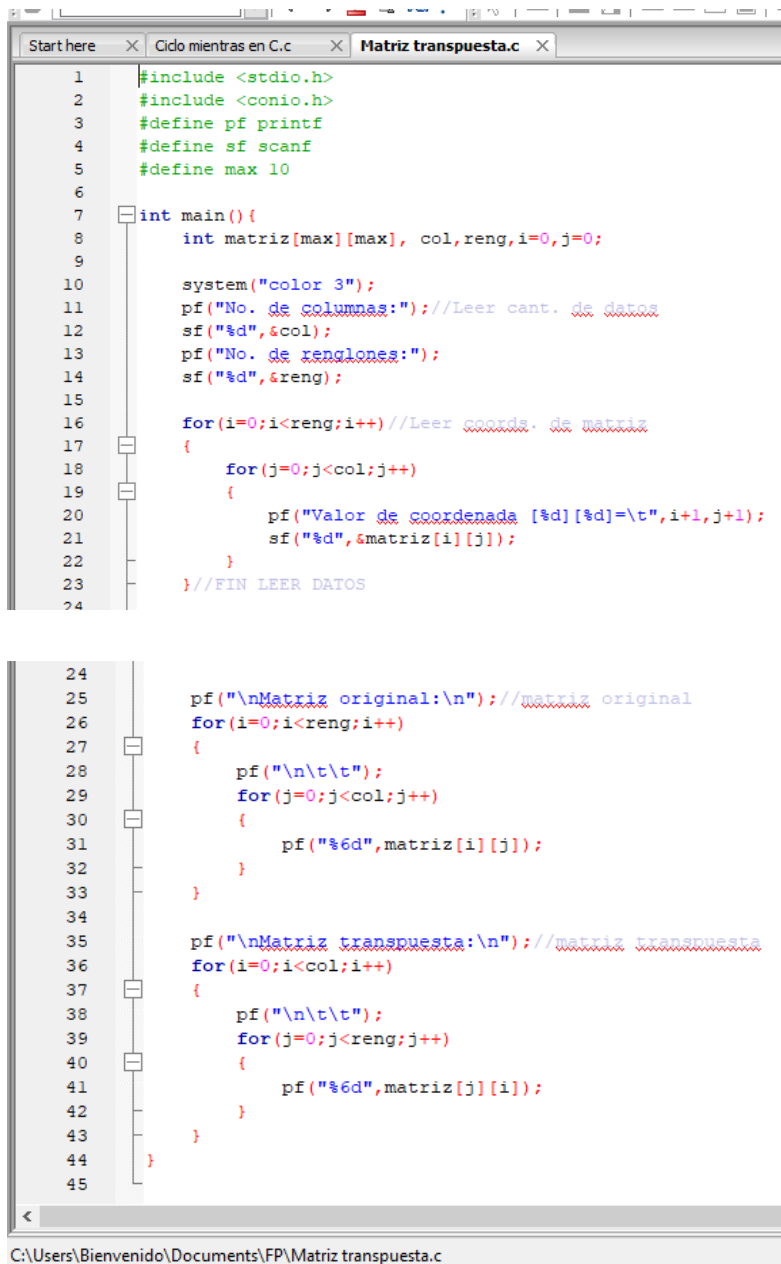


```
int arreglo[5];
```

Ese es un arreglo de 5 posiciones de números enteros.

## Parte IV.VIII. Matrices

Las matrices eran como los arreglos, se declaraban pero en lugar de un corchete, eran 2, representando las filas y columnas de una matriz cualquiera, a continuación pongo un ejemplo de un código en el que utilicé matrices:

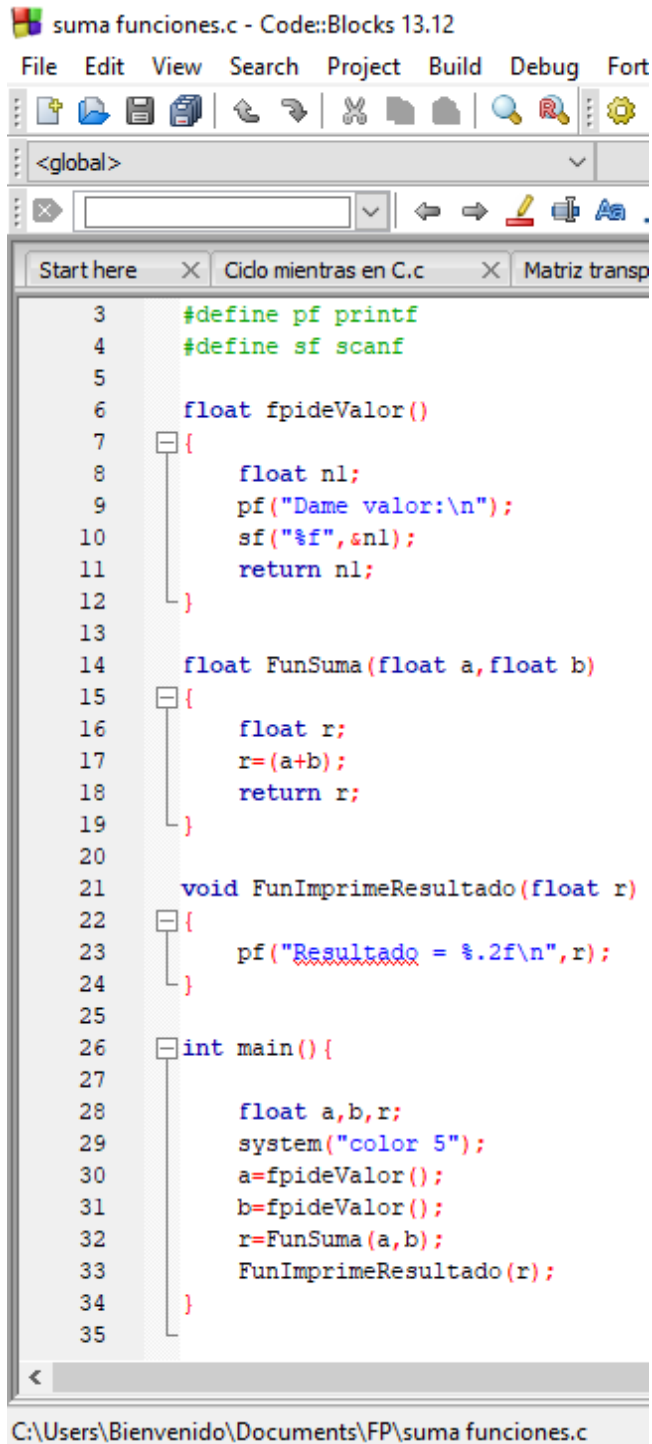


```
1  #include <stdio.h>
2  #include <conio.h>
3  #define pf printf
4  #define sf scanf
5  #define max 10
6
7  int main() {
8      int matriz[max][max], col, reng, i=0, j=0;
9
10     system("color 3");
11     pf("No. de columnas:"); //Leer cant. de datos
12     sf("%d", &col);
13     pf("No. de renglones:");
14     sf("%d", &reng);
15
16     for(i=0; i<reng; i++) //Leer datos de matriz
17     {
18         for(j=0; j<col; j++)
19         {
20             pf("Valor de coordenada [%d][%d]=\t", i+1, j+1);
21             sf("%d", &matriz[i][j]);
22         }
23     } //FIN LEER DATOS
24
25     pf("\nMatriz original:\n"); //matriz original
26     for(i=0; i<reng; i++)
27     {
28         pf("\n\t\t");
29         for(j=0; j<col; j++)
30         {
31             pf("%6d", matriz[i][j]);
32         }
33     }
34
35     pf("\nMatriz transpuesta:\n"); //matriz transpuesta
36     for(i=0; i<col; i++)
37     {
38         pf("\n\t\t");
39         for(j=0; j<reng; j++)
40         {
41             pf("%6d", matriz[j][i]);
42         }
43     }
44 }
45
```

C:\Users\Bienvenido\Documents\FP\Matriz transpuesta.c

## Parte IV. IX. Funciones

Las funciones nos sirven para acomodar mejor el código principal. Mientras que antes usábamos llaves para definir que entre ellas había una serie de acciones que se debían realizar, con las funciones podíamos meter todas las acciones en la función y en el programa principal simplemente llamar la función. Las funciones y su contenido van antes del programa principal, es decir, el programa principal quedaría al final de todo, si se quiere tener el programa principal primero y luego las funciones, entonces se deben declarar antes del programa principal y ya se utilizan cuando las necesites. Dejo un ejemplo:



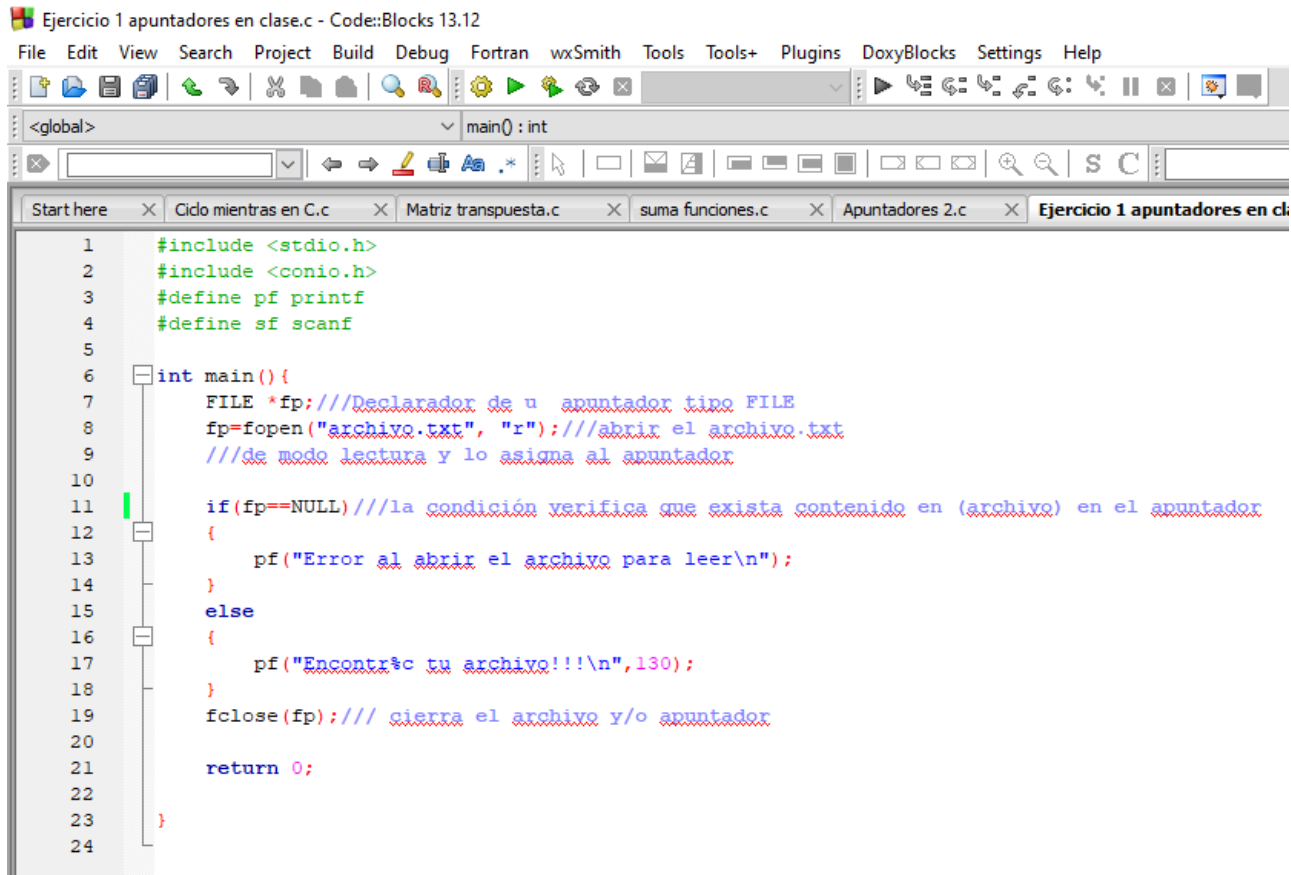
```
suma_funciones.c - Code::Blocks 13.12
File Edit View Search Project Build Debug Fort
<global>
Start here x Cido mientras en C.c x Matriz transp

3  #define pf printf
4  #define sf scanf
5
6  float fpideValor()
7  {
8      float nl;
9      pf("Dame valor:\n");
10     sf("%f",&nl);
11     return nl;
12 }
13
14 float FunSuma(float a,float b)
15 {
16     float r;
17     r=(a+b);
18     return r;
19 }
20
21 void FunImprimeResultado(float r)
22 {
23     pf("Resultado = %.2f\n",r);
24 }
25
26 int main(){
27
28     float a,b,r;
29     system("color 5");
30     a=fpideValor();
31     b=fpideValor();
32     r=FunSuma(a,b);
33     FunImprimeResultado(r);
34 }
35

C:\Users\Bienvenido\Documents\FP\suma_funciones.c
```

## Parte IV.X. Apuntadores

En esta parte vimos los apuntadores, que según lo que le entendí a maestra, son para mandar el contenido de X variable a otra o incluso a un documento en Word, una tabla de Excel o un bloc de notas, viceversa sólo lo hicimos de un bloc de notas a el ejecutable. A continuación presento un código que hicimos para ver este tema:



The screenshot shows a code editor window titled "Ejercicio 1 apuntadores en clase.c - Code::Blocks 13.12". The editor displays a C program that attempts to open a file named "archivo.txt" in read mode. The code includes standard headers and defines macros for printf and scanf. The main function checks if the file pointer is NULL and prints an error message if it is, or a success message if it is not. The code is as follows:

```
1  #include <stdio.h>
2  #include <conio.h>
3  #define pf printf
4  #define sf scanf
5
6  int main() {
7      FILE *fp; //Declarador de u apuntador tipo FILE
8      fp=fopen("archivo.txt", "r"); //abrir el archivo.txt
9      //de modo lectura y lo asigna al apuntador
10
11     if(fp==NULL) //la condición verifica que exista contenido en (archivo) en el apuntador
12     {
13         pf("Error al abrir el archivo para leer\n");
14     }
15     else
16     {
17         pf("Encontré tu archivo!!!\n", 130);
18     }
19     fclose(fp); // cierra el archivo y/o apuntador
20
21     return 0;
22 }
23
24
```

Como podemos ver, si no está previamente creado el archivo "archivo.txt", el programa diría que tuvo un error al encontrar el archivo, pero al crearlo no marca error.

## Parte IV.XI. Estructuras de datos

El último tema que vimos fue la estructura de datos, aquí lo que hacíamos es declarar una estructura con la instrucción "struct", le damos un nombre y dentro de ella podemos definir diferentes variables, aunque todas no sean del mismo tipo, al momento de querer guardar datos en esa variable con un scanf o si es una cadena con un gets, debemos llamar a la estructura y a su variable, a continuación muestro un ejemplo:

```
Estructuras de datos, n estudiantes.c - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plug
Start here x Ejercicio 1apuntadores en clase.c x Estructuras de datos ejemplo clase 2.c

1  #include <stdio.h>
2  #include <conio.h>
3  #define pf printf
4  #define sf scanf
5  int i,n;
6  struct datosAlumnos
7  {
8      char nombre[30];
9      char apellido[40];
10     char carrera[40];
11     char numCta[10];
12     int edad;
13     float estatura;
14     float peso;
15 };
16
17 struct datosAlumnos estudiante[3];
18
19 int main()
20 {
21     pf("%cCu%antos alumnos?\n",168,160);
22     sf("%i",&n);
23     for(i=0;i<n;i++)
24     {
25         fflush(stdin);
26         pf("Escribe el nombre del alumno\n");
27         gets(estudiante[i].nombre);
28         fflush(stdin);
29         pf("Escribe el apellido del alumno\n");
30         gets(estudiante[i].apellido);
31         fflush(stdin);
32         pf("Escribe la carrera del alumno\n");
33         gets(estudiante[i].carrera);
34         fflush(stdin);
35         pf("Escribe el n°mero de cuenta del alumno\n",163);
36         gets(estudiante[i].numCta);
37         fflush(stdin);
38         pf("Escribe la edad del alumno\n");
39         sf("%i",&estudiante[i].edad);
40         pf("Escribe la estatura del alumno\n");
41         sf("%f",&estudiante[i].estatura);
42         pf("Escribe el peso del alumno\n");
43         sf("%f",&estudiante[i].peso);
44     }
45     for(i=0;i<n;i++)
46     {
47         pf("Los datos del alumno %s %s con no. de cuenta %s son: \n",estudiante[i].nombre,estudiante[i].apellido,estudiante[i].numCta);
48         pf("Carrera: %s\n",estudiante[i].carrera);
49         pf("Edad: %i \n",estudiante[i].edad);
50         pf("Estatura: %f \n",estudiante[i].estatura);
51         pf("Peso: %f\n",estudiante[i].peso);
52     }
53 }
54
```

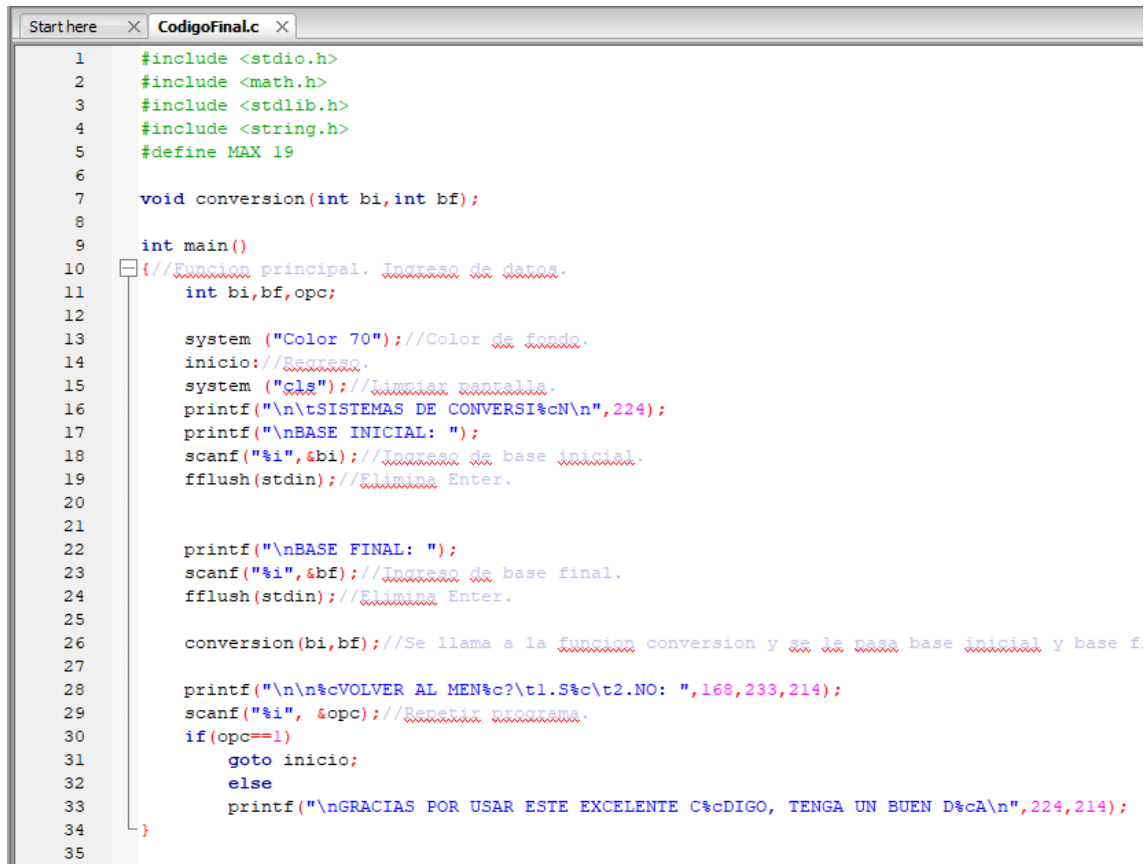
C:\Users\Bienvenido\Documents\FP\Estructuras de datos, n estudiantes.c

```
37     fflush(stdin);
38     pf("Escribe la edad del alumno\n");
39     sf("%i",&estudiante[i].edad);
40     pf("Escribe la estatura del alumno\n");
41     sf("%f",&estudiante[i].estatura);
42     pf("Escribe el peso del alumno\n");
43     sf("%f",&estudiante[i].peso);
44 }
45     for(i=0;i<n;i++)
46     {
47         pf("Los datos del alumno %s %s con no. de cuenta %s son: \n",estudiante[i].nombre,estudiante[i].apellido,estudiante[i].numCta);
48         pf("Carrera: %s\n",estudiante[i].carrera);
49         pf("Edad: %i \n",estudiante[i].edad);
50         pf("Estatura: %f \n",estudiante[i].estatura);
51         pf("Peso: %f\n",estudiante[i].peso);
52     }
53 }
54
```

C:\Users\Bienvenido\Documents\FP\Estructuras de datos, n estudiantes.c Windows (CR+LF) WINDOWS-1252 Line 1, Column 1 Insert Reac

## Parte V. Proyecto final

Ya para terminar el curso este trabajo, presento el proyecto final que hice en equipo: Un conversor de sistemas numéricos. El objetivo era hacer el conversor de los sistemas numéricos principales entre ellos, más aparte los base n, estos sistemas son el binario, decimal, octal y hexadecimal, más lo base n que son los base 3,4,5,6,7,9,etc. Dejo una imagen de nuestro código:

A screenshot of a code editor window titled 'CodigoFinal.c'. The code is a C program for converting numbers between different bases. It includes headers for stdio, math, stdlib, and string, and defines a constant MAX as 19. A function 'conversion' is declared. The 'main' function prompts the user for input base, output base, and an option to repeat. It uses 'system' calls for clearing the screen and setting color. It uses 'scanf' to get input and 'printf' for output. A loop is implemented using 'goto' to repeat the process if the user chooses. The code is color-coded with green for comments, blue for keywords, and black for identifiers and literals.

```
1  #include <stdio.h>
2  #include <math.h>
3  #include <stdlib.h>
4  #include <string.h>
5  #define MAX 19
6
7  void conversion(int bi,int bf);
8
9  int main()
10 { //Funcion principal. Ingreso de datos.
11     int bi,bf,opc;
12
13     system ("Color 70");//Color de fondo.
14     inicio://Regreso.
15     system ("cls");//Limpiar pantalla.
16     printf("\n\tSISTEMAS DE CONVERSIcN\n",224);
17     printf("\nBASE INICIAL: ");
18     scanf("%i",&bi);//Ingreso de base inicial.
19     fflush(stdin);//Elimina Enter.
20
21
22     printf("\nBASE FINAL: ");
23     scanf("%i",&bf);//Ingreso de base final.
24     fflush(stdin);//Elimina Enter.
25
26     conversion(bi,bf);//Se llama a la funcion conversion y se le pasa base inicial y base f.
27
28     printf("\n\n%cVOLVER AL MEN%c?\t1.S%c\t2.NO: ",168,233,214);
29     scanf("%i", &opc);//Repetir programa.
30     if(opc==1)
31         goto inicio;
32     else
33         printf("\n\nGRACIAS POR USAR ESTE EXCELENTE C%cDIGO, TENGA UN BUEN D%cA\n",224,214);
34 }
35
```

```

36 void conversion(int bi, int bf)
37 { //Funcion de conversion. Recibe a base inicial y base final.
38     int longi;
39     char numero[MAX];
40
41     printf("\nINTRODUCE UN NÚMERO: ", 233);
42     gets(numero); //gets guarda cada valor de la cadena de tipo char
43     fflush(stdin);
44
45     longi=strlen(numero); // saber la longitud del arreglo
46
47     if(longi>MAX) //Si longi es mayor a 19, ásea que el número introducido es muy largo, marca un error
48     {
49         printf("ERROR\n"); //error
50         exit(-1); //Termina el programa automáticamente
51     }
52
53     int i,y=0;
54     unsigned long long decimal=0; //Recibe el tipo de dato más grande.
55     int n[longi]; // un arreglo que depende la longitud del anterior.
56     char caracterActual;
57
58     for ( i = longi - 1; i >= 0; i--)
59     {
60         caracterActual = numero[i]; // Acá asigna el número a caracter actual en su código ascii
61
62         n[i]=caracterActual-48; //Resta 48 porque es 0 en ascii.
63
64         if (n[i]>16)
65         { //Si el número es mayor a 16 entonces
66             n[i]=n[i]-7; // entrar acá para convertir las letras a decimal
67         }
68         if (n[i]>bi-1||n[i]<0)
69         { //comprobar si un valor ingresado es menor o mayor a la base dada
70             printf("\nERROR\n");
71             exit(-1); //Si número ingresado no es parte de la base. sale.

```

```

65     { //Si el número es mayor a 16 entonces
66         n[i]=n[i]-7; // entrar acá para convertir las letras a decimal
67     }
68     if (n[i]>bi-1||n[i]<0)
69     { //comprobar si un valor ingresado es menor o mayor a la base dada
70         printf("\nERROR\n");
71         exit(-1); //Si número ingresado no es parte de la base. sale.
72     }
73     }
74
75     for(i=longi-1;i>=0;i--)
76     {
77         decimal+=(n[i]*(pow(bi,y))); // acá solo hace la conversión. Del número ingresado a decimal.
78         y++;
79     }
80
81     //////////////////////////////////////
82
83     int j, residuo;
84     i = 1;
85     while( decimal != 0 ) //Evalua si decimal es diferente a 0. Hace la operación a la base final.
86     {
87         residuo = decimal % bf ;
88         if( residuo < 10 ) //Si residuo es menor a 10, se le suma 48. Porque se trata de número en ascii.
89             residuo += 48 ;
90         else //Si el residuo es mayor o igual a 10, se le suma 55. Porque empiezan las letras en ascii.
91             residuo += 55 ;
92         n[ i++ ]= residuo; //Se le suma el número dependiendo de la condición anterior.
93         decimal /= bf; //Se hace la división para seguir evaluando.
94     }
95     printf( "\nEL NÚMERO CONVERTIDO ES: ", 233) ; //Imprime el arreglo que se almacena con la condición ant
96     for ( j = i -1 ; j > 0 ; j-- )
97         printf( "%c" , n[ j ] ) ;
98
99 }

```

El código, como se puede ver, nos quedó de un largo de 99 líneas, lo que hace es pasar de la base dada a decimal y de decimal a la base final, muy simple la verdad, no queríamos complicarnos la vida

haciendo formula de conversión por formula de conversión porque habría quedado un código muy largo.

## Bibliografía:

- Lucas, J. (2020, 10 septiembre). Qué es C: Características y sintaxis. OpenWebinars.net. <https://openwebinars.net/blog/que-es-c/>
- El código ASCII Completo, tabla con los codigos ASCII completos, caracteres simbolos letras ascii, ascii codigo, tabla ascii, codigos ascii, caracteres ascii, codigos, tabla, caracteres, simbolos, control, imprimibles, extendido, letras, vocales, signos, simbolos, mayusculas, minusculas, alt, teclas, acentos, agudo, grave, eñe, enie, arroba, dieresis, circunflejo, tilde, cedilla, anillo, libra, esterlina, centavo, teclado, tipear, escribir, español, ingles, notebook, laptop, asccii, asqui, askii, aski,20210225. (s. f.). El código ASCII Completo. Recuperado 25 de febrero de 2021, de <https://elcodigoascii.com.ar/>