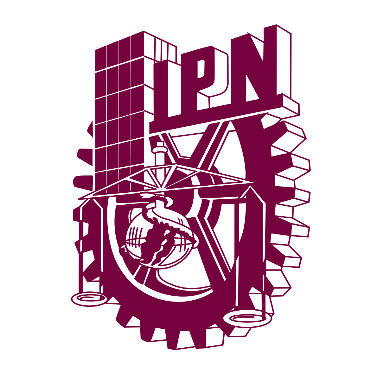
****

**Instituto Politécnico Nacional**

**Escuela Superior de Computo**

*Alumno:*

* *Monroy Ramírez Oscar G.*

***Grupo:*** *2CM23*

***Unidad de Aprendizaje:*** *Algoritmos y Estructuras de Datos*

***Evidencia:*** *Aritmética de Apuntadores*

***Docente:*** *De Luna Caballero Roberto*

***Fecha****: jueves, 15 de octubre de 2020*

Contenido

[Introducción 3](#_Toc53645775)

[Marco Teórico 4](#_Toc53645776)

[Definición de Apuntadores 4](#_Toc53645777)

[Apuntadores en aritmética. 4](#_Toc53645778)

[Declaración de apuntadores 5](#_Toc53645779)

[Asignación de apuntadores 5](#_Toc53645780)

[Des referenciación de apuntadores 5](#_Toc53645781)

[Verificación de tipos en apuntadores 5](#_Toc53645782)

[Matrices en C (Array unidimensional) 6](#_Toc53645783)

[Programa 7](#_Toc53645784)

[Ejemplo de Programa aritmética con Apuntadores 7](#_Toc53645785)

[Explicación del programa 15](#_Toc53645786)

[Programa central 15](#_Toc53645787)

[Función para la introducción de los datos 15](#_Toc53645788)

[Función de Suma 15](#_Toc53645789)

[Conclusiones 16](#_Toc53645790)

[Referencias Bibliográficas: 17](#_Toc53645791)

# 

# Introducción

En muchas ocasiones como programadores, nos podemos encontrar con diversos problemas, después de todo es parte fundamental darles solución, muchas veces son problemas los cuales son fáciles de resolver, y en otras ocasiones hay problemas más difíciles, en esta ocasión en la aritmética para los apuntadores, en si no es difícil, y se puede facilitar con el uso de apuntadores. El caso de las matrices es uno de los casos en los que es más fácil el uso de apuntadores, de hecho, es uno de los únicos métodos para introducir los datos y manejarlos.

También los apuntadores son muy necesarios para el manejo de datos entre funciones, pues es fundamental para no saturar de variables que solemos usar una o dos veces, y así podemos mover nuestro apuntador a diferentes ubicaciones, también es muy común el uso de este tipo de herramienta para el uso de memoria dinámica entre otros.

Concretamente para este problema, a pesar de que lo requerido era una matriz fija con datos proporcionados por el propio programa, siempre he sido de la idea que un programa finalmente es para un usuario, por lo que él debe decidir el tamaño y los valores con los cual es trabajar, pero también hay veces que el usuario sólo desea probar las capacidades del programa presentado, por ello este programa no excluye ninguna manera estás intenciones, dándole al usuario la capacidad de elegir el tamaño de su y de su contenido, o en caso de que lo prefiera una especie de demo que le permita ver cómo ópera el trabajo en sí.

El programa presenta muchas características requeridas para trabajar con apuntadores propiamente se usa el apuntador para indicar inicios y términos del arreglo que tenemos, y que utilizaremos muy a menudo para transportar de manera más fácil los datos a utilizar

es importante reconocer que dentro de mucho de lo que use usa en la programación, es claro que el dominio de los apuntadores es primordial, tanto como programadores, como para la facilidad y la comodidad a la hora de resolver y dar solución a un problema, que muchas veces nosotros mismos dificultamos al no saber operar este tipo de herramienta, por ello reconozco que es de vital importancia el trabajo aquí presentado, pues demuestra las capacidades y muchos de los usos que se le dan a los apuntadores, qué es esencialmente lo que se busca desarrollar, sin perder el tiempo en tecnicidades

Más adelante veremos cómo los apuntadores nos permiten de manera cómoda y objetiva trabajar con temas más avanzados como lo son las estructuras de datos y la memoria dinámica teniendo como principal objetivo alcanzar las habilidades, que permitan tener el dominio que requeriremos más adelante.

Por ello es una excelente forma de comenzar este curso…

# Marco Teórico

## Definición de Apuntadores

Según Kernighan (1978) cuando se declara una variable, el compilador reserva un espacio de memoria para ella y asocia el nombre de está a la dirección de memoria desde donde comienzan los datos de esa variable. Las direcciones de memoria se suelen describir como números en hexadecimal. Un apuntador es una variable cuyo valor es la dirección de memoria de otra variable. Se dice que un apuntador “apunta” a la variable cuyo valor se almacena a partir de la dirección de memoria que contiene el apuntador. Por ejemplo, si un apuntador p almacena la dirección de una variable x, se dice que “p apunta a x”. Características de los Apuntadores para aritmética

## Apuntadores en aritmética.

 Las operaciones válidas de apuntadores son (Kernighan):

* Asignación de apuntadores del mismo tipo.
* Adición de un apuntador y un entero.
* Sustracción de un apuntador y un entero.
* Resta de dos apuntadores a miembros del mismo arreglo.
* Comparación de dos apuntadores a miembros del mismo arreglo.
* Asignación del valor cero.
* Comparación con cero.

   Cualquier otra aritmética de apuntadores es ilegal; por lo tanto, no es legal sumar dos apuntadores, multiplicarlos o dividirlos, sumarles un valor **float**, o realizar asignaciones entre diferentes tipos de apuntadores sin una conversión de tipo forzada (*cast*).

Cuando se suma o resta un entero a un apuntador, dicho entero representa una *escala* que está en función del tamaño en *bytes* del tipo de datos a los que apunta el apuntador.

 Para visualizar mejor lo anterior, suponga que *ptr* es un apuntador a **int** que hace referencia al primer elemento de un arreglo de **int**, y que *n* es también un **int**, con *0 < n < TAM*(donde *TAM* representa el tamaño del arreglo), entonces la expresión:

ptr += n;

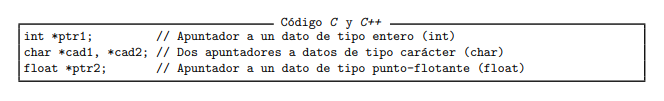
ó

ptr = ptr + n;

Indica al compilador que ptr se mueva o avance al n-ésimo elemento respecto de su posición actual.

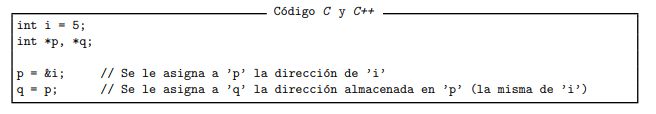
## Declaración de apuntadores

Para declarar un apuntador se especifica el tipo de dato al que apunta, el operador ‘\*’, y el nombre del apuntador. La sintaxis es la siguiente: \* A continuación se muestran varios ejemplos:



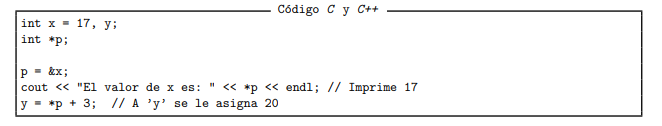
## Asignación de apuntadores

Se pueden asignar a un apuntador direcciones de variables a través del operador de referenciación (‘&’) o direcciones almacenadas en otros apuntadores. Ejemplos:



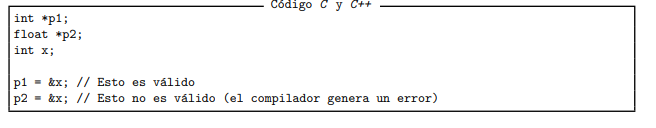
## Des referenciación de apuntadores

La de referenciación es la obtención del valor almacenado en el espacio de memoria donde apunta un apuntador. En C y C++ esto se hace a través del operador ‘\*’, aplicado al apuntador que contiene la dirección del valor. Nótese que se trata de un operador unario. Ejemplos:



## Verificación de tipos en apuntadores

Al igual que el resto de las variables, los apuntadores se enlazan a tipos de datos específicos (apuntadores a variables de cierto tipo), de manera que a un apuntador solo se le pueden asignar direcciones de variables del tipo especificado en la declaración del apuntador. Ejemplo:



## Matrices en C (Array unidimensional)

Según Hugo (2020):

Una matriz es un array multidimensional. Se definen igual que los vectores excepto que se requiere un índice por cada dimensión.

Una matriz es un vector de vectores o un también llamado array bidimensional. Una matriz bidimensional es una lista de matrices unidimensionales.

Su sintaxis es la siguiente:

Tipo nombre [tamaño 1][tamaño 2]...;

Una **matriz bidimensional** se podría representar gráficamente como una tabla con filas y columnas.

La **matriz tridimensional** se utiliza, por ejemplo, para trabajos gráficos con objetos 3D.

En el ejemplo puedes ver como se rellena y visualiza una matriz bidimensional. Se necesitan dos bucles para cada una de las operaciones. Un bucle controla las filas y otro las columnas.

EJEMPLO

/\* Matriz bidimensional. \*/

#include <stdio.h>

main() /\* Rellenamos una matriz \*/

{

int x,i,numeros[3][4];

/\* rellenamos la matriz \*/

for (x=0;x<3;x++)

for (i=0;i<4;i++)

scanf("%d",&numeros[x][i]);

/\* visualizamos la matriz \*/

for (x=0;x<3;x++)

for (i=0;i<4;i++)

printf("%d",numeros[x][i]);

}

Si al declarar una matriz también queremos inicializarla, habrá que tener encuenta el orden en el que los valores son asignados a los elementos de la matriz. Veamos algunos ejemplos:

int numeros[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};

Quedarían asignados de la siguiente manera:

numeros[0][0]=1 numeros[0][1]=2 numeros[0][2]=3 numeros[0][3]=4  
numeros[1][0]=5 numeros[1][1]=6 numeros[1][2]=7 numeros[1][3]=8  
numeros[2][0]=9 numeros[2][1]=10 numeros[2][2]=11 numeros[2][3]=12

También se pueden inicializar cadenas de texto:

char dias[7][10]={"lunes","martes","miércoles","jueves","viernes","sábado","domingo"};

Para referirnos a cada palabra bastaría con el primer índice:

printf("%s",dias[i]);

# Programa

El programa se ha adjuntado junto con el ejecutable correspondiente, sin embargo se anexa aquí de igual forma junto con explicación del programa y capturas de su uso

### Ejemplo de Programa aritmética con Apuntadores

#include <graphics.h>

#include <stdio.h>

#include <windows.h>

#include <time.h>

#include <dos.h>

#include <stdlib.h>

#include <unistd.h>

#include <string.h>

void MenuCentral();

void MenuDatos();

void MenuOperando();

void MenuOperando1();

int Imprimir(int\*, int\*, int, int, int);

int SumaCara(int\*, int\*, int, int, int);

int RestaCara(int\*, int\*, int, int, int);

int SumaColumna(int\*, int\*, int, int, int);

int SumaFila(int\*, int\*, int, int, int);

int RestaFila(int\*, int\*, int, int, int);

int RestaColumna(int\*, int\*, int, int, int);

int main(){

int \*ptr, \*ptl;

int caras, columnas, filas, menudatos, lectura, eleccion, operando, operando1, operando2, band=0;

MenuCentral();

scanf("%d",&lectura);

system("cls");

caras=lectura;

columnas=lectura;

filas=lectura;

int cubo[caras][filas][columnas];

do{

MenuDatos();

scanf("%d",&eleccion);

switch (eleccion){

case 1:{

printf("\nMuy bien, no es como que valga la pena meter los datos manualmente\n");

int i=0;

srand (time(NULL));

for (ptr=&cubo[0][0][0];ptr<=&cubo[caras-1][columnas-1][filas-1];ptr++, i++){

\*ptr = rand()%50;

i=i+1;

printf("Aleatorio Generado en Cubo posicion No.%d vale: %d\n",i,\*ptr);

i=i-1;

}

band=1;

break;

}

case 2:{

int i=0;

printf("\nExcelente, diviertete llenando el cubo!!!\n");

for (ptr=&cubo[0][0][0];ptr<=&cubo[caras-1][columnas-1][filas-1];ptr++, i++){

i=i+1;

printf("Introduzca el dato correspondiente No.%d:",i);

i=i-1;

scanf("%d",ptr);

}

band=1;

break;

}

default:{

printf("\nOpcion no valida, vuelva a intentar...\n\n");

}

}

}while (band==0);

system ("cls");

ptl=&cubo[0][0][0];

printf("\nContinuemos, ya hemos creado su cubo, veamos como se ve:\n");

Imprimir(ptl, ptr, caras, filas, columnas);

MenuOperando();

scanf("%d",&operando);

if (operando==1){

MenuOperando1();

scanf("%d",&operando1);

system("cls");

if(operando1==1){

SumaCara(ptl, ptr, caras, filas, columnas);

}

else if(operando1==2){

SumaFila(ptl, ptr, caras, filas, columnas);

}

else if(operando1==3){

SumaColumna(ptl, ptr, caras, filas, columnas);

}

else{

printf("Lo que desea ni esta en el menu\n");

}

}

else if (operando==2){

MenuOperando1();

scanf("%d",&operando1);

system("cls");

if(operando1==1){

RestaCara(ptl, ptr, caras, filas, columnas);

}

else if(operando1==2){

RestaFila(ptl, ptr, caras, filas, columnas);

}

else if(operando1==3){

RestaColumna(ptl, ptr, caras, filas, columnas);

}

else{

printf("\n\n\t\tUNA DISCULPA PERO NO ELIGIO UNA OPCION VALIDA, VUELVA A INTENTAR...");

}

}

else

printf("\n\n\t\tUNA DISCULPA PERO NO ELIGIO UNA OPCION VALIDA, VUELVA A INTENTAR...");

return 0;

}

void MenuCentral(){

system("cls");

printf("\t\t\t\t CALCULADORA DE CUBO\n");

printf("\nEste programa le ayudara a sumar, multiplicar o restar, los valores de un cubo generados aleatoriamente\n");

printf("\nDefina el tamaño de su cubo\n");

printf("\nElija el numero de caras, columnas y filas [n][n][n], introduzca n:");

printf("\n\n\n\t\t\t\t ELIGA SU OPCION:");

}

void MenuDatos(){

printf("\t\t\t\t ¿COMO DESEA MANEJAR SUS DATOS?\n");

printf("\n\nSeleccione la modalidad de insersion de datos\n\n");

printf("1.Generar Datos Aleatorios del 1 al 50\n");

printf("2.Yo tengo otros datos, y deseo introducirlos\n\n\n");

printf("\n\n\n\t\t\t\t ELIGA SU OPCION:");

}

void MenuOperando(){

system("cls");

printf("\t\t\t\t ¿AHORA COMO PROCEDEMOS?:");

printf("\n1.Procedamos con Sumas");

printf("\n2.Procedamos con Restas");

printf("\n\n\n\t\t\t\t ELIGA SU OPCION:");

}

void MenuOperando1(){

system("cls");

printf("\t\t\t\t ¿DONDE APLICAR LAS OPERACIONES?:");

printf("\n1.Caras");

printf("\n2.Filas");

printf("\n3.Columnas\n");

printf("\n\n\n\t\t\t\t ELIGA SU OPCION:");

}

int Imprimir(int \*ptl, int \*ptr, int caras, int filas, int columnas){ //aun esta en desarrollo, como veras no es funcional todavia...

system("cls");

int gd= DETECT, gm, cont=1, cont1=1, cont2=1, puntx=45, punty=10, Term, term, term1, Term1;

int \*ptt;

char \*ptb;

char bufer[caras\*filas\*columnas];

term=columnas\*25;

Term=filas\*35;

ptt=ptl;

ptb=bufer;

sprintf(ptb,"%d",\*ptt);

initwindow(3000,1080,"NOMBRE",0,0,false,true);

setlinestyle(3, 3, 3);

setcolor(1);

Term1=(40+Term);

setcolor(cont2);

rectangle(35,5,Term1,(term+10));

int x1=60+Term1, x2=160+Term1;

line(x1,80,x2,180);

line(x2,180,x2,380);

line(x2,380,x1,280);

line(x1,280,x1,80);

for(;ptt<ptr;ptt++,ptb++){

sprintf(ptb, "%d",\*ptt);

settextstyle(BOLD\_FONT, HORIZ\_DIR, 1);

outtextxy(puntx,punty,ptb);

if (cont==columnas){

punty=punty+25;

puntx=10;

cont=0;

}

if (cont1==(filas\*columnas)){

punty=punty+45;

Term1=40+Term;

term1=(punty)+term+5;

if (cont2!=caras){

setcolor(cont2+1);

rectangle(35,punty-5,Term1,term1);

x1=x1+30;

x2=x2+30;

line(x1,80,x2,180);

line(x2,180,x2,380);

line(x2,380,x1,280);

line(x1,280,x1,80);

}

puntx=10;

cont1=0;

cont2=cont2+1;

}

cont1=cont1+1;

cont=cont+1;

puntx=puntx+35;

}

delay(9999);

closegraph();

return 0;

}

int SumaCara(int \*ptl, int \*ptr, int caras, int filas, int columnas){

int sumatoria=0;

int contador=1;

int i=0;

int cara=1;

for(;ptl<=ptr;ptl++){

sumatoria=sumatoria+\*ptl;

if(contador==(filas\*columnas)){

printf("\nEste es el resultado de la Cara No. %d",cara);

printf("\n\t%d\n",sumatoria);

sumatoria=0;

contador=0;

cara=cara+1;

}

contador=contador+1;

i=i+1;

}

printf("\n\nGracias, hemos finalmente acabado",sumatoria);

}

int SumaFila(int \*ptl, int \*ptr, int caras, int filas, int columnas){

int sumatoria=0;

int contador=1;

int i=0;

int cara=1, fila=1;

for(;ptl<=ptr;ptl++){

sumatoria=sumatoria+\*ptl;

if(contador==(filas)){

printf("\nEste es el resultado de la Fila No. %d",fila);

printf("\n\t%d\n",sumatoria);

sumatoria=0;

contador=0;

fila=fila+1;

}

contador=contador+1;

}

printf("\n\nGracias, hemos finalmente acabado",sumatoria);

}

int SumaColumna(int \*ptl, int \*ptr, int caras, int filas, int columnas){

int sumatoria=0;

int contador=1;

int i=0;

int cara=1, columna=1;

for(;ptl<=ptr;ptl++){

sumatoria=sumatoria+\*ptl;

if(contador==(columnas)){

printf("\nEste es el resultado de la Columna No. %d",columna);

printf("\n\t%d\n",sumatoria);

sumatoria=0;

contador=0;

columna=columna+1;

}

contador=contador+1;

i=i+1;

}

printf("\n\nGracias, hemos finalmente acabado",sumatoria);

}

int RestaCara(int \*ptl, int \*ptr, int caras, int filas, int columnas){

int sumatoria=0;

int contador=1;

int i=0;

int cara=1;

for(;ptl<=ptr;ptl++){

sumatoria=sumatoria-\*ptl;

if(contador==(filas\*columnas)){

printf("\nEste es el resultado de la Cara No. %d",cara);

printf("\n\t%d\n",sumatoria);

sumatoria=0;

contador=0;

cara=cara+1;

}

contador=contador+1;

i=i+1;

}

printf("\n\nGracias, hemos finalmente acabado",sumatoria);

}

int RestaFila(int \*ptl, int \*ptr, int caras, int filas, int columnas){

int sumatoria=0;

int contador=1;

int i=0;

int cara=1, fila=1;

for(;ptl<=ptr;ptl++){

sumatoria=sumatoria-\*ptl;

if(contador==(filas)){

printf("\nEste es el resultado de la Fila No. %d",fila);

printf("\n\t%d\n",sumatoria);

sumatoria=0;

contador=0;

fila=fila+1;

}

contador=contador+1;

i=i+1;

}

printf("\n\nGracias, hemos finalmente acabado",sumatoria);

}

int RestaColumna(int \*ptl, int \*ptr, int caras, int filas, int columnas){

int sumatoria=0;

int contador=1;

int i=0;

int cara=1, columna=1;

for(;ptl<=ptr;ptl++){

sumatoria=sumatoria-\*ptl;

if(contador==(filas)){

printf("\nEste es el resultado de la Columna No. %d",columna);

printf("\n\t%d\n",sumatoria);

sumatoria=0;

contador=0;

columna=columna+1;

}

contador=contador+1;

i=i+1;

}

printf("\n\nGracias, hemos finalmente acabado",sumatoria);

}

## Explicación del programa

Para empezar el programa cuenta con 7 funciones las cuales 4 son para operaciones del contenido del cubo matricial una para introducir datos y para imprimirlos al igual un programa central desde donde se aloja toda la información relevante. Las 4 funciones para las operaciones y manejo de los contenidos del cubo son muy parecidas por lo cual solo se explicarán los referentes a la suma:

### Programa central

Este programa contiene toda la interfaz relevante para el usuario pues le permitirá elegir el tamaño de la matriz al igual que es donde se realiza la introducción de datos, que a su vez le permitirá al usuario decidir si quiere trabajar con datos generados al azar por el mismo programa, o trabajar con datos propios, posteriormente se pasa a la introducción de los datos según sea el caso y direcciona a la función correspondiente. Posteriormente se le muestra al usuario cuál es el contenido de la matriz que ha introducido, mostrándoselo de forma matricial y ordenada, para inmediatamente darle elegir qué tipo de operación desea hacer con ella, principalmente se le da a elegir entre operaciones de suma y resta, y ya especificado por el usuario qué tipo de operación trabajará si le pregunta de manera específica con qué parte del cubo desea hacer las operaciones dándole a elegir, entre caras filas y columnas.

### Función para la introducción de los datos

Esta es una de las partes del programa que más me gustan pues nos permite trabajar concretamente con la posibilidad de generar números completamente aleatorios para el arreglo, lo cual nos permite trabajar con nuevas funciones y específicamente nos permite darle la comodidad al usuario de elegir cómo trabajar el programa, sin que se tenga que acatar a números no deseados por el usuario y que pueda trabajar libremente con los que él desea probar él sinnúmero de funciones dentro del programa.

### Función de Impresión

En esta Si ocupan instrucciones del modo gráfico, qué mediante ciclos y apuntadores hacen una impresión de cada una de las caras y posteriormente ubica cada cara en un cubo digital, que te muestra según el color de cada una de las caras anteriormente empresas, la ubicación mediante el uso del mismo color en el cubo, lo que es importante recalcar es que esto es de vital importancia para visualizar los datos que se introdujeron se generaron anteriormente.

### Funciones de Operaciones

En lo que refiere a las operaciones todas ellas al igual que la introducción de datos dependen directamente de apuntadores que recorren la totalidad del cubo para su manejo, por ello es muy importante indicarle a las funciones el número de filas columnas y caras con las que se va a trabajar, todas las funciones realizan un conteo mientras el apuntador se va recorriendo dato por dato, para consultar la operación que se esté realizando justo en la terminación de filas columnas y caras, resultado práctico para programación y directamente resultando en como dos resultados, cada programa cuenta con la capacidad de imprimir al final de cada programa los resultados obtenidos, dando ordenados los datos como para la facilidad tanto visual como práctica del usuario.

# Conclusiones

El presente programa fue sin duda 1 de los programas en los que más trabajo me costó determinar cómo empezar , pues había muchos caminos que se podían tomar sin embargo estoy satisfecho con la elección que se realizó en cada 1 de los caminos con opciones a tomar, el más relevante de ellos fue el uso de las distintas funciones ocupadas en el programa aplicando así 4 de los temas más relevantes enseñados en el semestre anterior , por mencionarlos, tenemos el uso de argumentos, funciones, apuntadores, etc.

En con lo que respecta al uso de apuntadores consideró seriamente una de las herramientas más importantes a la hora de programar, pues nos permite evitarnos el cúmulo de variables basura, que lo único que hace es entorpecer muchas veces el manejo de la información albergada en ellas ,y es por ello que considero que es 1 de los recursos más valiosos para la construcción de estructuras de datos y qué será de gran ayuda en los aprendizajes y conocimientos por adquirir a lo largo de esta unidad de aprendizaje , por lo que entiendo totalmente la lógica de que sea 1 de los primeros programas a realizar en este curso punto fin.

# Referencias Bibliográficas:

**Delgado, Hugo.** (2020). Matrices en C - Bidimensional, tridimensional con ejemplos. Recuperado 27 de septiembre, 2020, de <https://disenowebakus.net/matrices.php>

Gil, M. A. (8 de 12 de 2004). *Univirsidad de Carobobo*. (L. Armando, Productor, & Univirsidad de Carobobo) Obtenido de https://utch2009.files.wordpress.com/2010/04/apuntadores.pdf

Brian W. K. & Dennis M. R. (1978) Lenguaje de programación C. Pearson Education Recuperado 29 septiembre, 2020 Disponible en:<http://www.tecnica1lomas.com.ar/tutoriales/lenguaje_C.pdf>

Ricardo Ruiz (2005) Introducción Datos Estruturados. Recuperado el 28 de septiembre,2020. Disponible en: <http://programacionestructuradarrr.blogspot.com/2016/09/aritmetica-de-apuntadores.html#:~:text=Las%20variables%20de%20tipo%20apuntador,sobre%20los%20valores%20que%20almacenan.>