



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

### Laboratorios de computación salas A y B

*Profesor:* **Manuel Enrique Castañeda Castañeda**

*Asignatura:*

**Estructura de Datos y Algoritmos I**

*Grupo:*

**12**

*No de Práctica(s):*

**04 . Almacenamiento en Tiempo de Ejecución**

*Integrante(s):*

- **Lemus Ambrosio Aline Andrea**
- **Reyes Fuentes José Manuel**
- **Sánchez Alvirde Andrés Iván**
- **Wong Sánchez Yibran Lee**

*No. de Equipo de cómputo  
empleado:*

**-**

*No. de Lista o Brigada:*

**12**

*Semestre:*

**2021-2**

*Fecha de entrega:*

**25 de junio de 2021**

*Observaciones:*

---

Calificación:

---

# **Estructura de Datos y Algoritmos I**

Práctica 04

Almacenamiento en Tiempo de  
Ejecución

## INTRODUCCIÓN

Durante esta práctica se pudo aplicar los conocimientos previos que pudimos ver durante las clases y de otras fuentes de información, como lo fueron páginas de internet y videos confiables que mostraban el uso de alguna función que se desconocía el funcionamiento, de igual forma para corregir algunos errores.

A lo largo de esta practica se estará utilizando conceptos ya antes vistos como lo fueron los apuntadores y arreglos, pero en esta práctica nos enfocaremos a hablar acerca de la memoria dinámica; aunque ya se habían utilizado en la anterior práctica hablaremos un poco más acerca de su uso y de cómo se deben declarar.

Para reservar la memoria estaremos utilizando la función “**malloc**” que especifica el número de bytes de memoria que el usuario quiere reservar y devuelve la dirección de memoria de la zona de memoria reservada. Con objeto de facilitar el "cálculo del número de bytes necesarios", el usuario puede especificar dicho número en función del tipo de dato que quiere reservar y del número de datos de dicho tipo.

Una manera correcta que podemos utilizar malloc sería la siguiente:

```
void *[puntero] = malloc([cantidad a reservar en bytes]);
```

Un ejemplo rápido de como se debe utilizar malloc para poder reservar la memoria sería de la siguiente manera:

```
#include <stdlib.h>

int main()
{
    void *p = malloc(5 * sizeof(int));
    int *_p = (int *) p;

    char *caracter = (char *) malloc(sizeof(char));
}
```

Una vez que hayamos hecho lo que sea con la memoria que hemos reservado con malloc(), si ya no la necesitamos conviene liberarla para que pueda ser utilizada posteriormente. Esto se consigue mediante la función free(), a la que pasamos como parámetro el puntero que deseamos liberar, borrando así la memoria dinámica que se le haya asignado anteriormente.

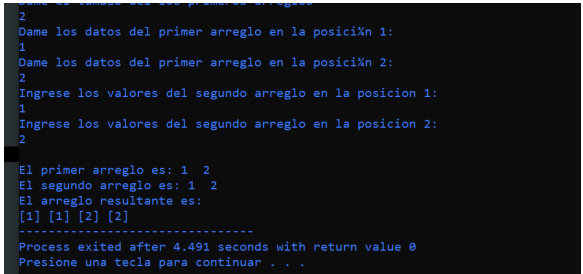
```
int main()
{
    int *entero = (int *) malloc(sizeof(int));
    comprobarMalloc(entero);

    //Más y más código...

    free(entero);
}
```

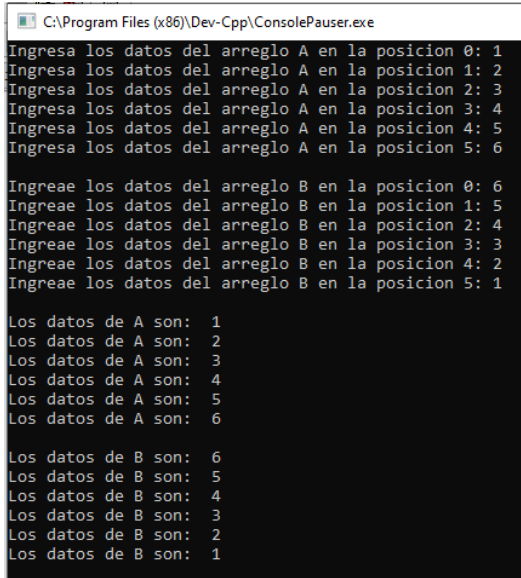
## DESARROLLO

3. genere un solo vector a partir de dos vectores iniciales, intercalando sus valores, utilizando aritmética de apuntadores.

CÓDIGO FUENTE	EJECUCIÓN
<pre>//genere un solo vector a partir de dos vectores iniciales, //intercalando sus valores, utilizando aritmética de apuntadores.  #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt;  int main() {     int i,n,p[n], s[n], t[2*n], aux,j,k;     j=0;     k=0;      printf("Dame el tamaño del los     primeros arreglos\n");     scanf("%d", &amp;n);     for(i=0;i&lt;n;i++)     {         printf("Dame los datos del         primer arreglo en la posición %d: \n",i+1     );         scanf("%d", &amp;p[i]);     }     for(i=0;i&lt;n;i++)     {         printf("Ingrese los valores         del segundo arreglo en la posicion %d:         \n",i+1);         scanf("%d", &amp;s[i]);     }     printf("\nEl primer arreglo es:");     for(i=0;i&lt;n;i++)     {         printf(" %d ",p[i]);     }     printf("\nEl segundo arreglo es:");     for(i=0;i&lt;n;i++)     {         printf(" %d ",s[i]);     } }</pre>	 <pre> 2 Dame el tamaño del los primeros arreglos: 1 Dame los datos del primer arreglo en la posición 1: 2 Ingrese los valores del segundo arreglo en la posición 1: 1 Ingrese los valores del segundo arreglo en la posición 2: 2  El primer arreglo es: 1 2 El segundo arreglo es: 1 2 El arreglo resultante es: [1] [1] [2] [2] ----- Process exited after 4.491 seconds with return value 0 Presione una tecla para continuar . . . </pre>

<pre>         }          //         for(i=0;i&lt;2*n;i++)         {             if(i%2==0)             {                 t[i]=p[j];                 j++;             }             else             {                 t[i]=s[k];                 k++;             }         }          printf("\nEl arreglo resultante es:\n");         for(i=0;i&lt;2*n;i++)         {             printf("[%i] ",t[i]);         }     } </pre>	
--	--

4. que intercambie los valores de dos arreglos

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio.h&gt; #define tam 6  int main(){      int A[tam], B[tam], i, temporal;     for( i=0; i&lt;tam; i++){         printf("Ingresa los datos del arreglo A en la posicion %d: ",i);         scanf("%d", &amp;A[i]);     }      printf("\n");     for( i=0; i&lt;tam; i++){         printf("Ingresa los datos del arreglo B en la posicion %d: ",i);         scanf("%d", &amp;B[i]);     } </pre>	 <pre> C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe Ingresa los datos del arreglo A en la posicion 0: 1 Ingresa los datos del arreglo A en la posicion 1: 2 Ingresa los datos del arreglo A en la posicion 2: 3 Ingresa los datos del arreglo A en la posicion 3: 4 Ingresa los datos del arreglo A en la posicion 4: 5 Ingresa los datos del arreglo A en la posicion 5: 6  Ingresa los datos del arreglo B en la posicion 0: 6 Ingresa los datos del arreglo B en la posicion 1: 5 Ingresa los datos del arreglo B en la posicion 2: 4 Ingresa los datos del arreglo B en la posicion 3: 3 Ingresa los datos del arreglo B en la posicion 4: 2 Ingresa los datos del arreglo B en la posicion 5: 1  Los datos de A son: 1 Los datos de A son: 2 Los datos de A son: 3 Los datos de A son: 4 Los datos de A son: 5 Los datos de A son: 6  Los datos de B son: 6 Los datos de B son: 5 Los datos de B son: 4 Los datos de B son: 3 Los datos de B son: 2 Los datos de B son: 1 </pre>

```

printf("\n");

for( i=0; i<tam; i++){
    printf("Los datos de A son:
%d\n",A[i]);
}

printf("\n");
for( i=0; i<tam; i++){
    printf("Los datos de B son:
%d\n",B[i]);
}

for( i=0; i<tam; i++){
    temporal=A[i];
    A[i]=B[i];
    B[i]=temporal;
}

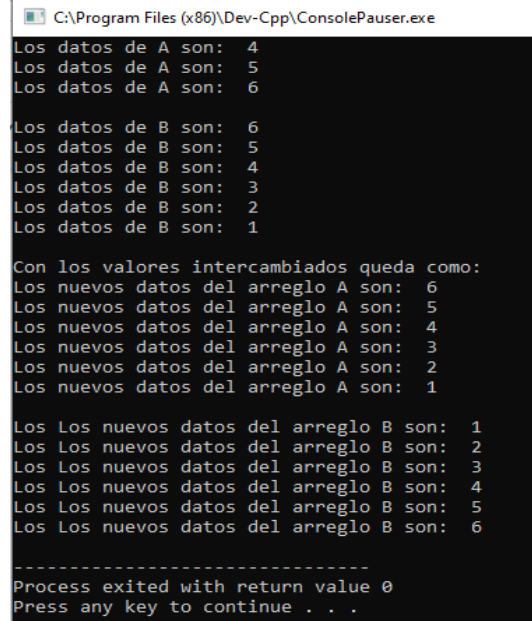
printf("\nCon los valores
intercambiados queda como:\n");

for( i=0; i<tam; i++){
    printf("Los nuevos datos del
arreglo A son: %d\n",A[i]);
}
printf("\n");

for( i=0; i<tam; i++){
    printf("Los Los nuevos datos del
arreglo B son: %d\n",B[i]);

}
}

```



```

C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe
Los datos de A son: 4
Los datos de A son: 5
Los datos de A son: 6

Los datos de B son: 6
Los datos de B son: 5
Los datos de B son: 4
Los datos de B son: 3
Los datos de B son: 2
Los datos de B son: 1

Con los valores intercambiados queda como:
Los nuevos datos del arreglo A son: 6
Los nuevos datos del arreglo A son: 5
Los nuevos datos del arreglo A son: 4
Los nuevos datos del arreglo A son: 3
Los nuevos datos del arreglo A son: 2
Los nuevos datos del arreglo A son: 1

Los Los nuevos datos del arreglo B son: 1
Los Los nuevos datos del arreglo B son: 2
Los Los nuevos datos del arreglo B son: 3
Los Los nuevos datos del arreglo B son: 4
Los Los nuevos datos del arreglo B son: 5
Los Los nuevos datos del arreglo B son: 6

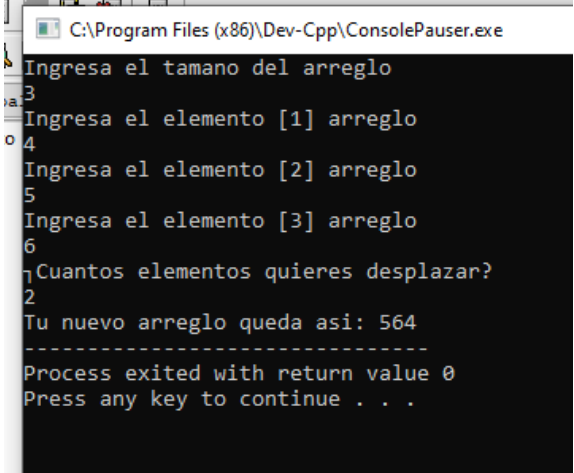
-----
Process exited with return value 0
Press any key to continue . . .

```

5. indique si una frase es un palíndromo, con aritmética de apuntadores

CÓDIGO FUENTE	EJECUCIÓN
<pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;string.h&gt; int main(){     int n,p=100;     char *palindromo;      palindromo=(char*)malloc(p*sizeof(char     *));     printf("Ingrese su palindromo\n");     fflush(stdin);     gets(palindromo);     n= strlen(palindromo);     for(int     i=0;i&lt;=strlen(palindromo);i++){      printf("%c\n",*(palindromo+i));     }     int a=0,k=0;     for(int     i=strlen(palindromo)-1;i&gt;=0;i--){      if(palindromo[i]==palindromo[a]){         k++;     }     a++;     }     if(strlen(palindromo)==k){         printf("\nEs         palindromo\n");     }     else{         printf("\nNo es         palindromo\n");     } }</pre>	

6. que con una función reciba un apuntador a un arreglo, su tamaño y un número que permita desplazar el arreglo hacia la derecha el número de posiciones que indique el número, de modo que los números de la derecha aparezcan por la izquierda.

CÓDIGO FUENTE	EJECUCIÓN
<pre> #include &lt;stdio&gt; #include &lt;stdlib&gt;  void recibir(); int *ap,a,n; int main(){     ap=(int*)malloc(a*sizeof(int));     recibir();  } void recibir() {     puts("Ingresa el tamaño del arreglo");     scanf("%d", &amp;a);     for(int i=0;i&lt;a;i++){         printf("Ingresa el elemento [%i] arreglo\n",i+1);         scanf("%d",&amp;ap[i]);     }      puts("¿Cuántos elementos quieres desplazar?");     scanf("%d", &amp;n);     int x[n];      if (a&lt;n){         puts("Datos no validos");     }      else {         for (int i=0;i&lt;=n;i++){             x[i]=ap[i];         }     }     printf("Tu nuevo arreglo queda asi: ");     for (int j=n-1;j&lt;a;j++) {         printf("%d",ap[j]);     }      for (int k=0;k&lt;n-1;k++){         printf("%d", x[k]);     } </pre>	 <pre> C:\Program Files (x86)\Dev-Cpp\ConsolePauser.exe Ingresa el tamaño del arreglo 3 Ingresa el elemento [1] arreglo 1 Ingresa el elemento [2] arreglo 2 Ingresa el elemento [3] arreglo 3 ¿Cuántos elementos quieres desplazar? 2 Tu nuevo arreglo queda asi: 564 ----- Process exited with return value 0 Press any key to continue . . . </pre>



}	
---	--

7. que utilice una función que devuelva el mayor, el menor y la media de los valores de un arreglo de números decimales, pasando los argumentos por referencia.

CÓDIGO FUENTE	EJECUCIÓN
<pre>//7. que utilice una función que devuelva el mayor, //el menor y la media de los valores de un //arreglo de números decimales, //pasando los argumentos por referencia.  #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt;  void ma(float *d, int m); void me(float *d, int m); void prom(float *d, int m);  int main() {     int n;     printf("%cDe que tamaño quieres tu arreglo?\n",168);     scanf("%d", &amp;n);     float *datos;     float dat=1;      datos=(float*)malloc(n*sizeof(float));     for(int i=0; i&lt;n;i++)     {         printf("Dame el dato del arreglo\n: ");         scanf("%f", &amp;dat);         datos[i]=dat;     }     system("cls");     for(int i=0;i&lt;n;i++)     {         printf(" %.1f ",datos[i]);     }      ma(datos,n);</pre>	<pre>1.0    2.0    3.0 El mayor es el 3.0  El menor es el 1.0 La media es: 2.0 Presione una tecla para continuar .</pre>

```

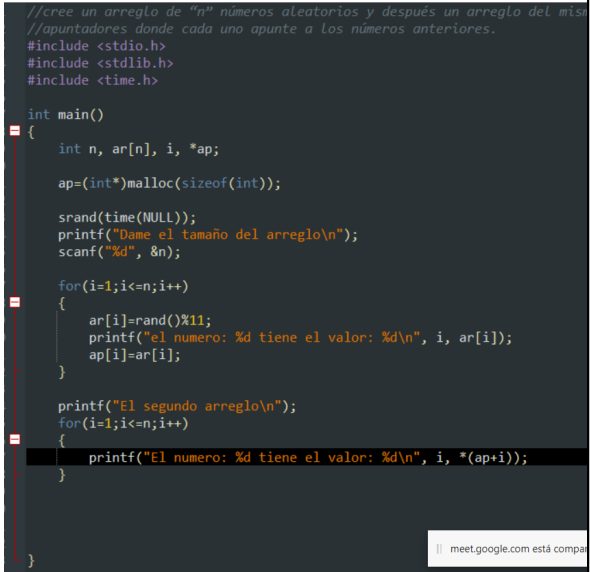
        me(datos,n);
        prom(datos,n);
        system("pause");
    }

    void ma(float *datos, int n)
    {
        float temp=0;
        for(int i=0; i<n;i++)
        {
            if(datos[i]>temp)
            {
temp=datos[i];
            }
        }
        printf("\n El mayor es el
%.1f\n", temp);
    }
    void me(float *datos, int n)
    {
        float temp=datos[0];
        for(int i=0;i<n;i++)
        {
            if(datos[i]<temp)
            {
temp=datos[i];
            }
        }
        printf("\n El menor es el
%.1f", temp);
    }

    void prom(float *datos, int n)
    {
        float temp=0;
        for(int i=0; i<n; i++)
        {
temp=temp+datos[i];
        }
        temp=(temp/n);
        printf("\nLa media es: %.1f
\n", temp);
    }

```

8. Que cree un arreglo de “n” números aleatorios y después un arreglo del mismo n apuntadores donde cada uno apunte a los números anteriores.

CÓDIGO FUENTE	EJECUCIÓN
<pre>//cree un arreglo de “n” números aleatorios y después un arreglo del mismo n //apuntadores donde cada uno apunte a los números anteriores. #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt;  int main() {     int n, ar[n], i, *ap;      ap=(int*)malloc(sizeof(int));      srand(time(NULL));     printf("Dame el tamaño del arreglo\n");     scanf("%d", &amp;n);      for(i=1;i&lt;=n;i++)     {         ar[i]=rand()%11;         printf("el numero: %d tiene el valor: %d\n", i, ar[i]);         ap[i]=ar[i];     }      printf("El segundo arreglo\n");     for(i=1;i&lt;=n;i++)     {         printf("El numero: %d tiene el valor: %d\n", i, *(ap+i));     }  }</pre>	 <pre>//cree un arreglo de “n” números aleatorios y después un arreglo del mismo n //apuntadores donde cada uno apunte a los números anteriores. #include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; #include &lt;time.h&gt;  int main() {     int n, ar[n], i, *ap;      ap=(int*)malloc(sizeof(int));      srand(time(NULL));     printf("Dame el tamaño del arreglo\n");     scanf("%d", &amp;n);      for(i=1;i&lt;=n;i++)     {         ar[i]=rand()%11;         printf("el numero: %d tiene el valor: %d\n", i, ar[i]);         ap[i]=ar[i];     }      printf("El segundo arreglo\n");     for(i=1;i&lt;=n;i++)     {         printf("El numero: %d tiene el valor: %d\n", i, *(ap+i));     }  }</pre>

9. que reciba la dirección de inicio de una cadena de caracteres y devuelve el número de vocales que hay en la cadena.

CÓDIGO FUENTE	EJECUCIÓN
---------------	-----------

```
#include <stdio.h>
#include <ctype.h>
```

```
int contarVocales(char *cadena);
```

```
int main(int argc, char const *argv[])
{
    char entrada[1000];
    printf("Escribe una frase:\n");
    gets(entrada);
    int vocales =
    contarVocales(entrada);
    printf("El numero de vocales que
    tiene tu frase es: %d\n", vocales);
    return 0;
}
```

```
int contarVocales(char *cadena){
    int vocales = 0;

    for (int indice = 0; cadena[indice]
    != '\0'; ++indice){

        char letraActual =
        tolower(cadena[indice]);

        if (
            letraActual == 'a' ||
            letraActual == 'e' ||
            letraActual == 'i' ||
            letraActual == 'o' ||
            letraActual == 'u'
        )
        {
            vocales++;
        }
    }
    return vocales;
}
```

```
Escribe una frase:
Hola mundo
El numero de vocales que tiene tu frase es: 4
```

### **OBSERVACIONES (Individuales)**

- **Lemus Ambrosio Aline Andrea**

La práctica de hoy se me hizo un poco complicada debido a que me llegue a confundir con los arreglos y en ocasiones a saber en donde estaban mis errores, gracias a la ayuda de mis compañeros y de lo que pude buscar para apoyarme en la resolución de la práctica, pude realizar mis ejercicios comprendiendo así mejor el tema que tratamos.

- **Reyes Fuentes José Manuel**

Realmente se me complico esta práctica debido a que algunos conceptos no los tenía tan claro y durante las clases algunos conceptos no quedaban del todo claro, que al final tuve que reforzarlo con mis compañeros de equipo y algunos videos encontrados en el internet fueron los que me ayudaron para que esta práctica no se me complica mucho.

Los conceptos que más tuve que batallar fueron los apuntadores como declararlos y básicamente el cómo usarlos a lo largo de los ejercicios.

- **Sánchez Alvirde Andrés Iván**

Esta práctica ha sido para mi complicada porque para empezar me confundia mucho lo que pedía el ejercicio y el cómo abordarlo, después de leerlo comprendí mejor lo que se tenía que hacer, pero aun así me atasque mucho en los ejercicios que tenían que ver con funciones y arreglos , también el ejercicio de reconocer un palíndromo, pero después de miles de intentos conseguí que detectara el palíndromo.

Me guíe en todo con ejercicios hechos en clase y con videos en internet.

- **Wong Sánchez Yibran Lee**

Esta práctica me resultó con alto grado de complejidad pero con la ayuda de mi equipo logramos resolver diversas problemáticas, en un principio repase diversos temas y conceptos que se llevaron a cabo en esta práctica, después de esto resultó más sencillo realizar los diversos programas.

## **CONCLUSIONES (Individuales)**

- **Lemus Ambrosio Aline Andrea**

Por medio de esta práctica comprendí mejor el funcionamiento de la función malloc, así como reforzar lo que habíamos visto acerca de punteros, si bien la práctica se me complicó por que no me acordaba bien de estas funciones, con las clases que tuvimos antes de la práctica, la ayuda del profesor y de mis compañeros pude recordar y aprender mejor las funciones que estaríamos usando, además de que estas me permitieron resolver mis ejercicios y en equipo poder completar la práctica de la mejor forma.

- **Reyes Fuentes José Manuel**

A lo largo de la práctica se tuvo que comprender algunas funciones y conceptos, en el caso de las funciones quedaron mucho más claras conforme las practica avanzaba quedaban mucho más claras.

Pero comparando con lo que fueron los apuntadores fue otro asunto debido a que fue demasiado difícil y frustrante ver que no compilaba, pero fui comprendiendo cómo se tenían que hacer y hasta quizás tomarle el gusto por usarlos, con esta enseñanza iré avanzando y comprendiendo mejor cómo hacerlos.

- **Sánchez Alvirde Andrés Iván**

Gracias a esta práctica maneje mucho mejor el uso de la memoria dinámica, el uso de los apuntadores y recordar cómo se tienen que usar las funciones a la hora de hacer los llamados y regresen lo que necesito.

Es bueno reforzar lo que ya se aprendió y pues con ello se hace ver lo útil que son el uso de apuntadores, funciones o memoria dinámica.

- **Wong Sánchez Yibran Lee**

Puedo concluir de esta práctica que el uso de memoria dinámica es muy útil para sintetizar código, también pudimos aprender a usar las funciones de malloc y que funciones tienen estos al momento de crear códigos, además de reforzar el conocimiento de las funciones dentro de los códigos, a pesar de poder tener complicaciones y errores en los códigos.