PRÁCTICA 1

1. VECTORES Y MATRICES (MEMORIA ESTÁTICA)

Objetivo: identificar usando vectores y matrices almacenados en memoria estática, la dimensión lógica y la dimensión física. Analizar espacio ocupado por dichas estructuras en memoria estática. Analizar espacio realmente utilizado por datos en dichas estructuras.

1) ¿Cuánto ocupará un arreglo de enteros de 100 elementos? ¿Y un arreglo de doubles de 100 elementos? Escriba y ejecute un programa que inicialice con valores aleatorios ambos arreglos, los muestre por pantalla y por último muestre la cantidad de bytes que ocupa cada uno. Implemente con funciones. Utilice la función `sizeof()` cuando sea necesario.

2) Escriba y ejecute un programa que lea 2 enteros. Estos dos enteros definen las dimensiones de una matriz (filas, columnas). La matriz tiene como dimensión física NxN, y dimensión lógica filasxcolumnas (enteros leídos). El programa debe inicializar la matriz con valores aleatorios entre 20 y 30. Luego, se debe imprimir la matriz de dos formas: por filas y por columnas. Definir funciones para inicializar y mostrar la matriz. Explique: ¿qué sucede si N es muy grande? ¿Qué sucede si N es muy pequeño?

3) Escriba y ejecute un programa que lea un entero. Este entero define la dimensión lógica de una matriz cuadrada (filas = columnas) de enteros. Escriba una función que inicialice una matriz identidad (matriz identidad: m[i][j] = 1 si i = j, m[i][j] = 0 caso contrario). Escriba otra función que reciba la matriz y su dimensión lógica e imprima sus valores. Asumir que la matriz es de N x N como máximo (dimensión física). Si N = 500, calcule la cantidad de memoria que se reserva para la matriz y cuánto se usa realmente si se leen los valores: 10, 100, 50. Responda para cada valor: ¿cuántos bytes tienen datos válidos y cuántos no?

4) Escriba y ejecute un programa que declare una matriz de enteros. Asuma una matriz cuadrada y varíe sus dimensiones. Pruebe distintos valores como: 100, 500, 1000... hasta que dé un error en el momento de ejecución. ¿Por qué sucede esto?

5) Escriba y ejecute un programa que mantenga información de números complejos (usando el tipo de datos definido en el trabajo práctico 0). Se desea tener 2 matrices de complejos (de 10 filas por 10 columnas) y realizar la suma y resta de dichas matrices. Implemente con funciones.

6) Escribir y ejecutar un programa que inicialice un arreglo de N enteros y luego elimine la priner ocurrencia de un número ingresado por parámetro. La eliminación debe ser tal que si el número no existe en el arreglo, el arreglo queda de dimensión N. Si el elemento existe, el arreglo debe quedar de N-1 enteros y la eliminación genera un corrimiento pisando el elemento eliminado: si se elimina el elemento de la posición 3, se debe mover el ekenebti de posición 4 a la posición 3, el elementoo de la posición 5 a la posición 4 y así sucesivamente, reescribiendo las posiciones de los datos.

7) Dado un arreglo de 1000 enteros, realizar una función que reciba dicho arreglo y un número cualquiera (ingresado por teclado). Dicha función debe informar si el entero se encuentra en el vector. Realizar esta función de dos maneras distintas:

1. Recorriendo el vector de enteros en su totalidad.
2. Recorriendo hasta que encuentre el dato (o se termine el vector).

Conteste: ¿qué solución cree que es mejor? ¿por qué? con la segunda opción: ¿cuántas veces cómo mínimo se comparan valores? ¿y cómo máximo? ¿y promedio?

2. PUNTEROS

8) Indicar el tipo de cada una de las siguientes variables:

```
int *pt_x, *dir_y, x, y;
long *dir_dt, *dir_pt;
double a, b, c;
```

9) Dado el siguiente código, realizar un diagrama donde se muestre por cada instrucción lo que sucede con los contenidos de las variables, suponiendo que la variable “a” se encuentra en la direccón 5858 y la variable “b” en la dirección de memoria 9000.

```
int *dir_a, *dir_b;
int  a, b;

a = 3;
dir_a = &a;
dir_b = &b;
*dir_b = *dir_a;
```

10) Escriba un programa en C que incluya las siguientes instrucciones de declaración:

```
int num, cuenta;
long fecha;
float pendiente;
double potencia;
```

a) Utilice el operador de dirección (&) y la función `printf()` para mostrar las direcciones correspondientes a cada variable. b) Utilice la función `sizeof()` para determinar el tamaño de cada variable. c) Al código anterior, agregue las siguientes declaraciones:

```
int *p_num;
long *p_fecha;
float *p_pendiente;
double *p_potencia;
```

¿Qué imprime cada una de las siguientes líneas? Explique en cada caso lo que imprime y justifique.

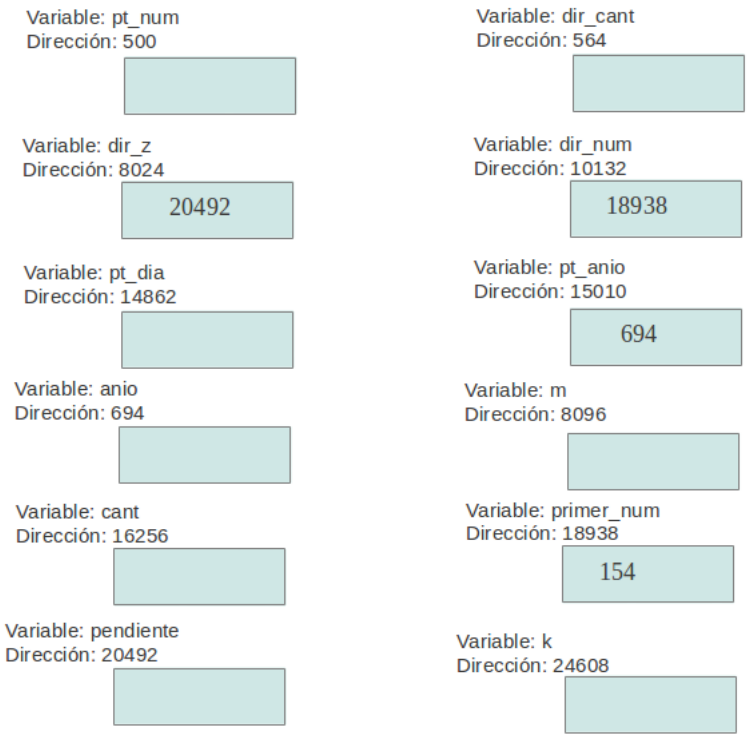
```
printf("Cantidad de bytes de 'p_num': %ld y contenido de p_num %ld \n",
       sizeof(p_num), sizeof(*p_num));

printf("Cantidad de bytes de 'p_potencia': %ld y contenido de p_potencia:
       %ld \n", sizeof(p_potencia), sizeof(*p_potencia));

printf("Cantidad de bytes de 'p_pendiente': %ld y contenido de
       p_pendiente: %ld \n", sizeof(p_pendiente), sizeof(*p_pendiente));
```

11) Para las siguientes variables y direcciones ilustradas en la figura, escriba los datos apropiados determinados por las siguientes instrucciones

```
a. pt_num = &m;
b. dir_cant = &cant;
c. *dir_z = 25;
d. k = *dir_num;
e. pt_dia = dir_z;
f. *pt_anyo = 1987;
g. *dir_cant = *dir_num;
```



12) ¿Qué significa la siguiente declaración? Grafique la variable `ptr`.

```
int *ptr[10];
```

A partir de la declaración anterior y asumiendo que edad1 y edad2 son dos variables enteras. ¿Es válido el siguiente código?

```
ptr[5] = &edad1;
ptr[6] = &edad2;
```

Agregue estas asignaciones al gráfico.

13) Explique: ¿cómo haría para que una función escrita en C retorne más de un valor? Utilizar la bibliografía y citar las referencias.

14) Considere el pasaje de parámetros por valor y por referencia. Determinar qué imprime el siguiente programa:

Algorithm 1: Pasaje de parámetros.

```
1  #include <stdio.h>
2  int funcion1(int x, int y)
3  {
4      x = x + 1;
5      y = y - 1;
6      printf("Dentro de funcion1, x = %d, y = %d ", x, y);
7      return 0;
8  }
9  int funcion2(int *x, int *y)
10 {
11     *x = *x + 1;
12     *y = *y - 1;
13     printf("Dentro de funcion1, x = %d, y = %d ", *x, *y);
14     return 0;
15 }
16 int main()
17 {
18     int a, b;
19     a = 3;
20     b = 10;
21     printf(“Antes de llamar a funcion1, a = %d, b = %d ", a, b);
22     funcion1(a,b);
23     printf("Despues de llamar a funcion1, a = %d, b = %d ", a, b);
24     printf(“Antes de llamar a funcion2, a = %d, b = %d ", a, b);
25     funcion2(&a,&b);
26     printf("Despues de llamar a funcion2, a = %d, b = %d ", a, b);
27     return 0;
28 }
```

¿Qué es lo que se está pasando como argumente en el llamado:

```
funcion2(&a, &b);
```

¿Qué es lo que se especifica en el encabezamiento de la `funcion2()` con respecto a los parámetros llamados x e y?

15) Implemente la función `exchangeInt()` entre 2 variables enteras. Implemente la función exchangeCom() entre dos complejos.

3. ALGORITMOS DE ORDENACIÓN Y BÚSQUEDA

16) ¿Cuál es la diferencia entre ordenación por selección y ordenación por inserción?

17) Un vector contiene los elementos mostrados a continuación. Los primeros dos elementos se han ordeado utilizando un algoritmo de inserción. ¿Cuál será el valor de los elementos del vector después de tres pasadas más del algoritmo?

3	13	8	25	45	23	98	58

18) Dada la siguiente lista:

47	3	21	32	56	92

Después de dos pasadas de un algoritmo de ordenación, el arreglo ha quedado dispuesto así:

3	21	47	32	56	92

¿Qué algoritmo de ordeanción se está utilizando (selección, burbuja o inserción)? Justifique la respuesta.

19) Un arreglo contiene los elementos indicados más abajo. Utilizando el agoritmo de ordenación Shell encuentre las pasadas y los intercambios que se realizan para su ordenación.

8	43	17	6	40	16	18	97	11	7

20) Un arreglo contiene los elementos indicados a continuación. Utilizando el algoritmo de búsqueda binaria, trazar las etapas necesarias para encontrar el número 88.

8	13	17	26	44	56	88	97

Igual búsqueda pero para el número 20.

21) Escribir y ejecutar un programa que ordene un vector de 50 enteros. Primero se debe mostrar los datos en orden creciente y luego en orden decreciente. Utilizar el método Burbuja.

22) Se realiza una encuesta anónima sobre 50 personas, donde de cada una se conoce la edad y la cantidad de hijos. Luego de ingresar los datos, se procesan los datos para obtener: 1. El listado de los 50 participantes, ordenados por edad e informando edad y cantidad de hijos. Utilizar el método de inserción para realizar la ordenación. 2. Un listado donde se informe el promedio de hijos para cada una de las edades consideradas (establecer como pre-condición el rango de edades que se usará).

23) Escribir y ejecutar un programa que mantenga información referente a coches en una ciudad. Sólo se dispone de matrícula y de modelo (simplificar y asumir que ambos son números enteros). A partir de esta información, generar 2 vectores: uno ordenado de forma creciente por matrícula el otro ordenado de forma decreciente

por modelo. Mostrar los vectores resultantes. Utilizar el método de selección para realizar la ordenación.

24) Se tienen 2 vectores A y B, ambos de 20 enteros. Se pide ordenar ambos vectores de forma creciente y luego, generar un vector C de 40 enteros que sea el "merge" de dichos vectores A y B. La operación de merge consiste en intercalar los datos de los vectores A y B de forma ordenada en el vector C: cada dato insertado en C es el mínimo entre A y B (y se avanza en el vector donde se encontraba el mínimo). Utilizar el método `shellsort` para realizar la ordenación de los vectores A y B. Mostrar el vector C ordenado.