PRÁCTICA 1

Objetivo: repaso de arreglos en memoria estática. Repaso de punteros. Presentación y uso de algoritmos de búsqueda y de ordenación.

1. VECTORES Y MATRICES (MEMORIA ESTÁTICA)

1) Resuelva en equipos de 3 integrantes las funciones pedidas para operar con vectores y con matrices (memoria estática). Su equipo debe resolver la totalidad de las funciones. El código debe compilar y ejecutar de forma correcta las funciones pedidas. En el aula virtual de la asignatura se encuentran los templates (plantillas) de los códigos a resolver.

2. PUNTEROS

2) 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;
```

3) 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;
```

4) 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.

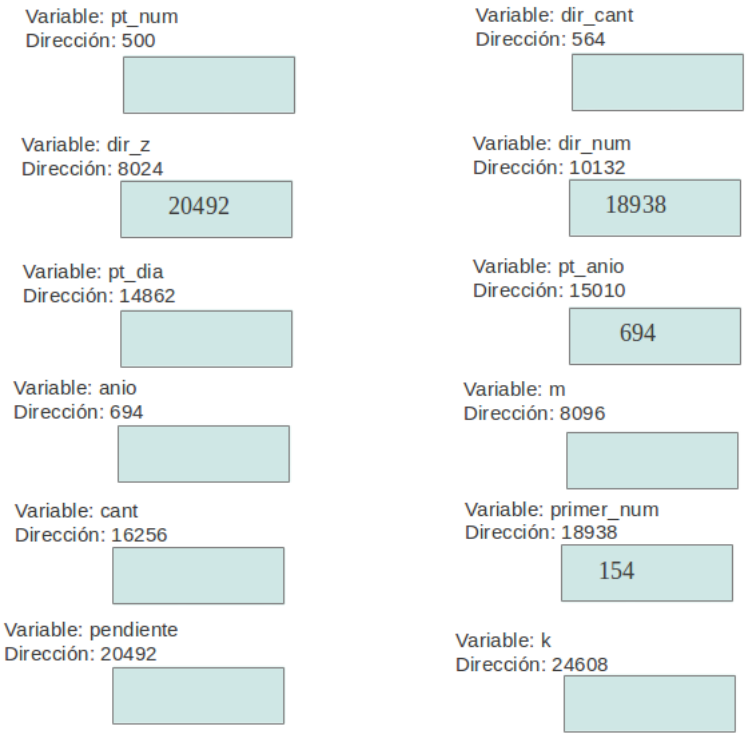
```
printf("Cantidad de bytes de 'p_num': %ld y contenido de p_num %ld \n",
       sizeof(p_num), sizeof(*p_num));

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));
```

5) Para las siguientes variables y direcciones ilustradas en la figura, escriba los valores correctos al ejecutar 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;
```



6) ¿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.

7) Explique: ¿cómo haría para que una función escrita en C retorne más de un valor?

8) 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?

9) Implemente la función `swapInt()` entre 2 variables enteras. Dicha función intercambia los valores de los parámetros. Implemente la función swapCom() entre dos complejos.

3. ALGORITMOS DE ORDENACIÓN Y BÚSQUEDA

15) 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.

16) Se realiza una encuesta anónima sobre 50 personas, donde de cada persona se conoce la edad y la cantidad de hijos. Luego de ingresar los datos, los mismos se procesan 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).

17) Escribir y ejecutar un programa que mantenga información referente a coches en una ciudad. Sólo se dispone de número de patente 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 patente y el otro ordenado de forma decreciente por modelo. Mostrar los vectores resultantes. Utilizar el método de selección para realizar la ordenación.

18) 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.