Trabajo Práctico N° 1 – Arreglos

Consideraciones para la resolución de la guía:

1- Para cada uno de los problemas propuestos deberá completar:
   - Fase de resolución de problemas.
     - Análisis del problema:
       - Identificación de datos de entradas.
       - Identificación de resultados de salidas.
       - Tareas o procesos
     - Diseño del algoritmo.
       - Pseudocódigo.
       - Diagrama de Flujo
       - Verificación del diseño.
       - Prueba de escritorio (utilizar caso de prueba especificado en el enunciado).
   - Fase de implementación
     - Codificación del algoritmo – Lenguaje de programación Python 3.6.
     - Compilación y ejecución del programa.
       - Identificar errores si existen (compilación y ejecución).
       - Corregir errores.
       - Compilar y ejecutar nuevamente.
     - Verificación.
       - Usar mismos casos de prueba realizados en la prueba de escritorio.
       - Comparar los resultados de la prueba de escritorio con los de la ejecución.
     - Depuración.
       - Identificar errores si existen (compilación, ejecución, lógicos)
       - Corregir los errores.
       - Verificar el caso de prueba nuevamente.

2- Fase de resolución del problema: Los archivos generados con PSeInt (*.psc)* correspondientes a los diseños en pseudocódigo y diagrama de flujo de sus soluciones propuestas, deben tener la siguiente estructura:

```
1   //Ejercicio N° X
2   //
3   //FASE DE RESOLUCION DEL PROBLEMA
4   //ANALISIS DEL PROBLEMA
5   //Resultados o salidas:
6   //Datos o entradas:
7   //Procesos:
8   //
9   //DISEÑO DEL ALGORITMO - PSEUDOCODIGO
10
11  Algoritmo titulo_disenio
12      //Diseño del algoritmo
13
14  FinAlgoritmo
```

3- Fase de implementación: los archivos generados con el IDE Geany (*.py)* con el código fuente en lenguaje Python de sus soluciones propuestas deben tener la siguiente estructura:

```
#Ejercicio N X
#FASE DE IMPLEMENTACION
#Escritura del programa

print("Codifico mi diseño de algoritmo en Python")
```

Es importante mencionar que en los cuestionarios teórico-prácticos y parciales se incluirán ejercicios similares a los propuestos en esta guía por lo que te recomendamos que para practicar intentes resolverlos a TODOS.

Ejercicios propuestos

1. Diseñar un programa que permita cargar dos arreglos unidimensionales llamados *A* y *B,* ambos de *n* elementos enteros. Luego deben mostrar los elementos de estos arreglos y generar un tercer vector llamado *C,* del mismo tamaño *n*. Cada elemento de *C* será igual a la sumatoria del elemento del vector *A* y del vector *B* en la misma posición. Antes de finalizar se debe mostrar: el vector generado *C,* el promedio del vector, el acumulado de la sumatoria de sus elementos, el elemento de menor valor y el elemento de mayor valor. Se aclara que el valor de *n* es solicitado al usuario al inicio.

Lote de prueba N° 1:

Entrada:

```
Cuantos elementos tendrán los vectores ? 5

Comienzo de carga del vector A...
Elemento  0
10
Elemento  1
25
Elemento  2
34
Elemento  3
52
Elemento  4
20

Comienzo de carga del vector B...
Elemento  0
20
Elemento  1
40
Elemento  2
54
Elemento  3
77
Elemento  4
35
```

Salida:

```
Vector A: 10 25 34 52 20
Vector B: 20 40 54 77 35

Generando el vector C...
Vector C generado...

Vector C: 30 65 88 129 55
Acumulado del vector C: 367
Promedio del vector C: 73.4
Mayor elemento del vector C: 129
Menor elemento del vector C: 30
```

Lote de prueba N° 2:

Entrada:

```
Cuantos elementos tendrán los vectores ? 6

Comienzo de carga del vector A...
Elemento  0
33
Elemento  1
41
Elemento  2
29
Elemento  3
68
Elemento  4
47
Elemento  5
3

Comienzo de carga del vector B...
Elemento  0
11
Elemento  1
69
Elemento  2
74
Elemento  3
55
Elemento  4
83
Elemento  5
91
```

Salida:

```
Vector A: 33 41 29 68 47 3
Vector B: 11 69 74 55 83 91

Generando el vector C...
Vector C generado...


Vector C: 44 110 103 123 130 94
Acumulado del vector C: 604
Promedio del vector C: 100.67
Mayor elemento del vector C: 130
Menor elemento del vector C: 44
```

Lote de prueba N° 3:

Entrada:

```
Cuantos elementos tendrán los vectores ? 9

Comienzo de carga del vector A...
Elemento  0
25
Elemento  1
11
Elemento  2
4
```

```
Elemento  3
7
Elemento  4
20
Elemento  5
35
Elemento  6
9
Elemento  7
12
Elemento  8
2

Comienzo de carga del vector B...
Elemento  0
3
Elemento  1
7
Elemento  2
55
Elemento  3
24
Elemento  4
81
Elemento  5
20
Elemento  6
10
Elemento  7
3
Elemento  8
44
```

Salida:

```
Vector A: 25 11 4 7 20 35 9 12 2
Vector B: 3 7 55 24 81 20 10 3 44

Generando el vector C...
Vector C generado...

Vector C: 28 18 59 31 101 55 19 15 46
Acumulado del vector C: 372
Promedio del vector C: 41.33
Mayor elemento del vector C: 101
Menor elemento del vector C: 15
```

2. El gerente de las salas de cine *Cinemax* desea tener en claro algunos números sobre las películas más vistas. El cine dispone de un total de cinco (5) películas en cartelera por semana y siete (7) salas para proyectarlas. Cualquier sala puede proyectar cualquier película en diferentes horarios.

Se solicita diseñar un programa que permita cargar la cantidad de espectadores que han asistido a ver cada película en cada sala por semana para luego determinar:

- Total de espectadores que asistieron al cine.
- Mejor combinación sala-película.

- Película más vista.
- Total de espectadores que asistieron a cada sala durante las 5 películas. El total de espectadores de cada sala se debe cargar en un vector llamado *totalesSalas*. Mostrar el vector resultante.
- Total de espectadores que vieron cada película en las 7 salas. El total de espectadores que vieron cada película se debe cargar en un vector llamado *totalesPelis*. Mostrar el vector resultante.

Nota: implementar lo solicitado por medio de funciones.

Lote de prueba N° 1:

Entrada:

```
Sala 1 - P1: 23 | P2: 21 | P3: 26 | P4: 53 | P 5: 65 |
Sala 2 - P1: 45 | P2: 73 | P3: 1  | P4: 64 | P 5: 4  |
Sala 3 - P1: 50 | P2: 96 | P3: 11 | P4: 66 | P 5: 81 |
Sala 4 - P1: 4  | P2: 16 | P3: 13 | P4: 88 | P 5: 46 |
Sala 5 - P1: 33 | P2: 77 | P3: 94 | P4: 92 | P 5: 17 |
Sala 6 - P1: 81 | P2: 97 | P3: 68 | P4: 53 | P 5: 33 |
Sala 7 - P1: 35 | P2: 81 | P3: 79 | P4: 76 | P 5: 17 |
```

Salida:

```
Totales de espectadores en el cine: 1779

Mejor combinación sala/película
Sala N° 6 - Película N° 2
Cantidad de espectadores: 97

Película más vista - Película N° 4
Cantidad de espectadores totales: 492

Totales de espectadores por sala
Sala N° 1: 188 | Sala N° 2: 187 | Sala N° 3: 304 | Sala
N° 4: 167 | Sala N° 5: 313 | Sala N° 6: 332 | Sala N°
7: 288 |

Totales de espectadores por película
Película N° 1: 271 | Película N° 2: 461 | Película N°
3: 292 | Película N° 4: 492 | Película N° 5: 263 |
```

Lote de prueba N° 2:

Entrada:

```
Sala 1 - P1: 80 | P2: 14 | P3: 74 | P4: 1  | P5: 87 |
Sala 2 - P1: 16 | P2: 48 | P3: 68 | P4: 31 | P5: 82 |
Sala 3 - P1: 55 | P2: 1  | P3: 63 | P4: 87 | P5: 99 |
Sala 4 - P1: 0  | P2: 65 | P3: 71 | P4: 93 | P5: 60 |
Sala 5 - P1: 11 | P2: 19 | P3: 0  | P4: 55 | P5: 65 |
Sala 6 - P1: 5  | P2: 3  | P3: 41 | P4: 5  | P5: 32 |
Sala 7 - P1: 6  | P2: 28 | P3: 64 | P4: 53 | P5: 58 |
```

Salida:

```
Totales de espectadores en el cine: 1540

Mejor combinación sala/película
Sala N° 3 - Película N° 5
```

```
Cantidad de espectadores: 99

Película más vista - Película N° 5
Cantidad de espectadores totales: 483

Totales de espectadores por sala
Sala N° 1: 256 | Sala N° 2: 245 | Sala N° 3: 305 | Sala
N° 4: 289 | Sala N° 5: 150 | Sala N° 6: 86 | Sala N°
7: 209 |

Totales de espectadores por película
Película N° 1: 173 | Película N° 2: 178 | Película N°
3:381 | Película N° 4: 325 | Película N° 5: 483 |
```

Lote de prueba N° 3:

Entrada:

```
Sala 1 - P1: 77 | P2: 9 | P3: 30 | P4: 83 | P5: 58 |
Sala 2 - P1: 89 | P2: 59 | P3: 22 | P4: 13 | P5: 62 |
Sala 3 - P1: 39 | P2: 81 | P3: 61 | P4: 75 | P5: 86 |
Sala 4 - P1: 49 | P2: 26 | P3: 33 | P4: 83 | P5: 15 |
Sala 5 - P1: 92 | P2: 68 | P3: 71 | P4: 58 | P5: 10 |
Sala 6 - P1: 28 | P2: 10 | P3: 70 | P4: 5 | P5: 36 |
Sala 7 - P1: 88 | P2: 99 | P3: 58 | P4: 23 | P5: 54 |
```

Salida:

```
Totales de espectadores en el cine: 1820

Mejor combinación sala/película
Sala N° 7 - Película N° 2
Cantidad de espectadores: 99

Película más vista - Película N° 1
Cantidad de espectadores totales: 462

Totales de espectadores por sala
Sala N° 1: 257 | Sala N° 2: 245 | Sala N° 3: 342 | Sala
N° 4: 206 | Sala N° 5: 299 | Sala N° 6: 149 | Sala N°
7: 322 |

Totales de espectadores por película
Película N° 1: 462 | Película N° 2: 352 | Película N°
3:345 | Película N° 4: 340 | Película N° 5: 321 |
```

3. Diseñar un programa que lea un número de NIS correspondiente a una conexión de electricidad y lo busque en un vector llamado *pendientesLuz* de *20* posiciones. Este arreglo mantiene en memoria los números de NIS de todas aquellas conexiones pendientes de realizar. Para simular la funcionalidad requerida, el arreglo puede ser cargado de manera aleatoria (valores aleatorios comprendidos en el intervalo [1, 50000].

Antes de finalizar mostrar el vector *pendientesLuz* desordenado, luego mostrarlo ordenado y con un mensaje en pantalla tal como ‘NIS con conexión de luz pendiente’ en caso de encontrar el NIS ingresado, o ‘NIS ya conectado’ en caso de no encontrarlo. La búsqueda debe ser binaria, queda a criterio del programador el algoritmo de ordenamiento a implementar.

Dado que el arreglo de números NIS será generado de manera aleatoria, los lotes de prueba son solo a modo de ejemplo.

Lote de prueba N° 1:

Entrada:

```
Ingrese el nro. de NIS: 18580
```

Salida:

```
NIS con conexiones pendientes - Desordenado
15952|26567|8248|16403|3563|16730|8338|25872|1768
9|24740|20097|10528|15246|12436|21254|5077|4837|2
5451|4654|18580|

NIS con conexiones pendientes - Ordenado
3563|4654|4837|5077|8248|8338|10528|12436|15246|1
5952|16403|16730|17689|18580|20097|21254|24740|25
451|25872|26567|

NIS con conexión de luz pendiente
```

Lote de prueba N° 2:

Entrada:

```
Ingrese el nro. de NIS: 10354
```

Salida:

```
NIS con conexiones pendientes - Desordenado
1713|7980|20038|26572|13678|4572|12488|27345|7676
|12182|19623|17991|21407|14053|15701|25259|11369|
14045|24997|16208|

NIS con conexiones pendientes - Ordenado
1713|4572|7676|7980|11369|12182|12488|13678|14045
|14053|15701|16208|17991|19623|20038|21407|24997|
25259|26572|27345|

NIS ya conectado
```

A partir de ahora los ejercicios no incluyen los lotes de prueba correspondientes. Con lo practicado hasta el momento, asumimos que ya se encuentran en condiciones de confeccionar sus propios lotes de prueba para verificar los diseños de los algoritmos y programas a través de la prueba de escritorio.

Recordemos que, los lotes de prueba, son las posibles situaciones en cuanto a datos de entrada que el programa tendrá que procesar para entregar los resultados o salidas esperadas. Dependiendo del tipo de problema se pueden presentar 2 o más lotes de prueba. Es recomendable considerar un escenario normal y uno o más casos extremos. Debemos probar con distintos valores para determinar si los controles que ponemos están funcionando correctamente., como por ejemplo: no ingresar una fecha de nacimiento que sea mayor que la fecha actual, no ingresar un número negativo donde deber ir uno positivo o sin decimales, no ingresar un valor numérico donde solo debe ir texto, no ingresar un valor fuera del rango establecido, etc.

4. Desarrollar un programa que permita crear en memoria tres matrices llamadas *marco, diagP* y *diagS* de longitud 10x10. Los tres arreglos creados solo deberán ser rellenados con los valores numéricos 0 y 1 pero no de cualquier manera:
   - Matriz *marco:* el valor 1 (uno) ocupará las posiciones o elementos que delimitan la tabla, es decir, el borde o marco de la tabla, mientras que el resto de los elementos contendrán el valor 0 (cero).
   - Matriz *diagP:* el valor 1 (uno) ocupará la diagonal principal mientras que el resto de los elementos contendrán el valor 0 (cero).
   - Matriz *diagS:* el valor 1 (uno) ocupará la diagonal secundaria o contradiagonal mientras que el resto de los elementos contendrán el valor 0 (cero).

Lote de prueba N° 1:

```
Entrada:
      No hay entradas
Salida:
      Matriz marco
      1111111111
      1000000001
      1000000001
      1000000001
      1000000001
      1000000001
      1000000001
      1000000001
      1000000001
      1111111111

      Matriz diagP
      1000000000
      0100000000
      0010000000
      0001000000
      0000100000
      0000010000
      0000001000
      0000000100
```

```
0000000010
0000000001

Matriz diagS
0000000001
0000000010
0000000100
0000001000
0000010000
0000100000
0001000000
0010000000
0100000000
1000000000
```

5. Simule un juego de lotería para 50 jugadores, utilizando como estructura de datos en memoria, un arreglo unidimensional llamado *jugadores*. Cada posición del arreglo representa el número correspondiente a la jugada de un jugador que será cargado utilizando una función llamada *jugada()*. El número ingresado debe estar comprendido en el intervalo [0, 100]. Se aclara que para agilizar la carga del arreglo se debe rellenar con valores aleatorios.

Luego de ingresar todas las jugadas, el programa debe simular el juego a través de una función llamada *lotería()*. Ésta función genera al azar un número aleatorio, también comprendido en el intervalo [0, 100], y determina si hubo algún ganador entre los *n* jugadores (o más de uno) informado la situación con un mensaje en pantalla.

Dado que el arreglo de jugadas y el número sorteado son generados de manera aleatoria, los lotes de prueba son solo a modo de ejemplo.

Lote de prueba N° 1:

Entrada:

```
Bienvenido al juego de la lotería

Por favor jugadores, ingresen sus jugadas...
Nro. de lotería - Jugador N° 1: 70
Nro. de lotería - Jugador N° 2: 35
Nro. de lotería - Jugador N° 3: 80
Nro. de lotería - Jugador N° 4: 97
Nro. de lotería - Jugador N° 5: 71
Nro. de lotería - Jugador N° 6: 81
Nro. de lotería - Jugador N° 7: 65
Nro. de lotería - Jugador N° 8: 8
Nro. de lotería - Jugador N° 9: 67
Nro. de lotería - Jugador N° 10: 30
Nro. de lotería - Jugador N° 11: 2
Nro. de lotería - Jugador N° 12: 90
Nro. de lotería - Jugador N° 13: 72
Nro. de lotería - Jugador N° 14: 46
Nro. de lotería - Jugador N° 15: 7
Nro. de lotería - Jugador N° 16: 22
Nro. de lotería - Jugador N° 17: 1
Nro. de lotería - Jugador N° 18: 97
```

```
Nro. de lotería - Jugador N° 19: 47
Nro. de lotería - Jugador N° 20: 39
Nro. de lotería - Jugador N° 21: 4
Nro. de lotería - Jugador N° 22: 93
Nro. de lotería - Jugador N° 23: 100
Nro. de lotería - Jugador N° 24: 53
Nro. de lotería - Jugador N° 25: 92
Nro. de lotería - Jugador N° 26: 4
Nro. de lotería - Jugador N° 27: 54
Nro. de lotería - Jugador N° 28: 2
Nro. de lotería - Jugador N° 29: 23
Nro. de lotería - Jugador N° 30: 22
Nro. de lotería - Jugador N° 31: 87
Nro. de lotería - Jugador N° 32: 88
Nro. de lotería - Jugador N° 33: 75
Nro. de lotería - Jugador N° 34: 64
Nro. de lotería - Jugador N° 35: 20
Nro. de lotería - Jugador N° 36: 62
Nro. de lotería - Jugador N° 37: 80
Nro. de lotería - Jugador N° 38: 89
Nro. de lotería - Jugador N° 39: 43
Nro. de lotería - Jugador N° 40: 13
Nro. de lotería - Jugador N° 41: 31
Nro. de lotería - Jugador N° 42: 55
Nro. de lotería - Jugador N° 43: 48
Nro. de lotería - Jugador N° 44: 79
Nro. de lotería - Jugador N° 45: 61
Nro. de lotería - Jugador N° 46: 33
Nro. de lotería - Jugador N° 47: 1
Nro. de lotería - Jugador N° 48: 80
Nro. de lotería - Jugador N° 49: 15
Nro. de lotería - Jugador N° 50: 78
```

Salida:

```
Jugador 7 ganaste la lotería
Numero sorteado: 65
Cantidad de ganadores: 1
```

Lote de prueba N° 2:

Entrada:

```
Bienvenido al juego de la lotería

Por favor jugadores, ingresen sus jugadas...
Nro. de lotería - Jugador N° 1: 79
Nro. de lotería - Jugador N° 2: 52
Nro. de lotería - Jugador N° 3: 65
Nro. de lotería - Jugador N° 4: 55
Nro. de lotería - Jugador N° 5: 45
Nro. de lotería - Jugador N° 6: 76
Nro. de lotería - Jugador N° 7: 61
Nro. de lotería - Jugador N° 8: 49
Nro. de lotería - Jugador N° 9: 31
Nro. de lotería - Jugador N° 10: 5
Nro. de lotería - Jugador N° 11: 78
Nro. de lotería - Jugador N° 12: 84
Nro. de lotería - Jugador N° 13: 21
```

```
Nro. de lotería - Jugador N° 14: 89
Nro. de lotería - Jugador N° 15: 26
Nro. de lotería - Jugador N° 16: 7
Nro. de lotería - Jugador N° 17: 21
Nro. de lotería - Jugador N° 18: 35
Nro. de lotería - Jugador N° 19: 48
Nro. de lotería - Jugador N° 20: 45
Nro. de lotería - Jugador N° 21: 6
Nro. de lotería - Jugador N° 22: 51
Nro. de lotería - Jugador N° 23: 37
Nro. de lotería - Jugador N° 24: 26
Nro. de lotería - Jugador N° 25: 66
Nro. de lotería - Jugador N° 26: 66
Nro. de lotería - Jugador N° 27: 12
Nro. de lotería - Jugador N° 28: 47
Nro. de lotería - Jugador N° 29: 53
Nro. de lotería - Jugador N° 30: 75
Nro. de lotería - Jugador N° 31: 20
Nro. de lotería - Jugador N° 32: 42
Nro. de lotería - Jugador N° 33: 81
Nro. de lotería - Jugador N° 34: 80
Nro. de lotería - Jugador N° 35: 45
Nro. de lotería - Jugador N° 36: 1
Nro. de lotería - Jugador N° 37: 78
Nro. de lotería - Jugador N° 38: 69
Nro. de lotería - Jugador N° 39: 96
Nro. de lotería - Jugador N° 40: 51
Nro. de lotería - Jugador N° 41: 78
Nro. de lotería - Jugador N° 42: 77
Nro. de lotería - Jugador N° 43: 14
Nro. de lotería - Jugador N° 44: 13
Nro. de lotería - Jugador N° 45: 17
Nro. de lotería - Jugador N° 46: 51
Nro. de lotería - Jugador N° 47: 79
Nro. de lotería - Jugador N° 48: 28
Nro. de lotería - Jugador N° 49: 61
Nro. de lotería - Jugador N° 50: 65
```

Salida:

```
Numero sorteado: 59
No hubo ganadores
```

Lote de prueba N° 3:

Entrada:

```
Bienvenido al juego de la lotería

Por favor jugadores, ingresen sus jugadas...
Nro. de lotería - Jugador N° 1 : 96
Nro. de lotería - Jugador N° 2 : 29
Nro. de lotería - Jugador N° 3 : 96
Nro. de lotería - Jugador N° 4 : 2
Nro. de lotería - Jugador N° 5 : 95
Nro. de lotería - Jugador N° 6 : 24
Nro. de lotería - Jugador N° 7 : 8
Nro. de lotería - Jugador N° 8 : 78
```

```
Nro. de lotería - Jugador N° 9 : 92
Nro. de lotería - Jugador N° 10 : 4
Nro. de lotería - Jugador N° 11 : 27
Nro. de lotería - Jugador N° 12 : 84
Nro. de lotería - Jugador N° 13 : 38
Nro. de lotería - Jugador N° 14 : 23
Nro. de lotería - Jugador N° 15 : 11
Nro. de lotería - Jugador N° 16 : 86
Nro. de lotería - Jugador N° 17 : 35
Nro. de lotería - Jugador N° 18 : 28
Nro. de lotería - Jugador N° 19 : 56
Nro. de lotería - Jugador N° 20 : 49
Nro. de lotería - Jugador N° 21 : 57
Nro. de lotería - Jugador N° 22 : 9
Nro. de lotería - Jugador N° 23 : 57
Nro. de lotería - Jugador N° 24 : 87
Nro. de lotería - Jugador N° 25 : 95
Nro. de lotería - Jugador N° 26 : 6
Nro. de lotería - Jugador N° 27 : 86
Nro. de lotería - Jugador N° 28 : 89
Nro. de lotería - Jugador N° 29 : 98
Nro. de lotería - Jugador N° 30 : 10
Nro. de lotería - Jugador N° 31 : 49
Nro. de lotería - Jugador N° 32 : 29
Nro. de lotería - Jugador N° 33 : 51
Nro. de lotería - Jugador N° 34 : 9
Nro. de lotería - Jugador N° 35 : 32
Nro. de lotería - Jugador N° 36 : 24
Nro. de lotería - Jugador N° 37 : 22
Nro. de lotería - Jugador N° 38 : 100
Nro. de lotería - Jugador N° 39 : 92
Nro. de lotería - Jugador N° 40 : 26
Nro. de lotería - Jugador N° 41 : 29
Nro. de lotería - Jugador N° 42 : 66
Nro. de lotería - Jugador N° 43 : 60
Nro. de lotería - Jugador N° 44 : 84
Nro. de lotería - Jugador N° 45 : 94
Nro. de lotería - Jugador N° 46 : 27
Nro. de lotería - Jugador N° 47 : 83
Nro. de lotería - Jugador N° 48 : 14
Nro. de lotería - Jugador N° 49 : 99
Nro. de lotería - Jugador N° 50 : 61
```

Salida:

```
Jugador 20 ganaste la lotería
Jugador 31 ganaste la lotería
Numero sorteado: 49
Cantidad de ganadores: 2
```

6. Diseñar un programa que permita cargar un arreglo unidimensional con 5 elementos de tipo entero y luego lo muestre por pantalla.

7. Diseñar un programa que permita cargar un vector con los números pares del 10 al 30 inclusive y luego los imprima en pantalla del último al primero.

8. Desarrollar un programa que permita cargar por filas un arreglo bidimensional de 3x4 elementos enteros comprendidos en el intervalo [0, 50] para luego mostrarlo en pantalla.

9. Se solicita diseñar un programa que permita cargar por columnas un arreglo bidimensional de 3x4 elementos enteros comprendidos en el intervalo [0, 9] para luego mostrarlo en pantalla. Además debe calcular la sumatoria de cada una de las filas y el producto de cada una de las columnas.

10. Diseñar un programa que permita crear en memoria un arreglo unidimensional con una cantidad *n* de elementos enteros. La cantidad *n* de elementos del vector se solicita al inicio al usuario. Luego, por medio de una función llamada *cargarArray(),* rellenar el arreglo con los múltiplos de otro número entero *k* también solicitado al usuario. Recordemos que un número dado es múltiplo de otro si le contiene un número entero de veces. Por ejemplo, si se definió un arreglo con *n* igual a 5 elementos y el número *k* elegido en la función fue 3 (tres), el vector contendrá los cinco primeros múltiplos de tres: 3, 6, 9, 12 y 15. Antes de finalizar mostrar los valores del arreglo en pantalla por medio de otra función llamada *mostrarArray()*.

11. Desarrollar un programa que permita cargar un vector de 20 elementos enteros. Durante la carga se deberá tener en cuenta que los valores dentro del arreglo no pueden repetirse. Antes de finalizar mostrar el vector con los valores finales.

12. Desarrollar un programa que permita crear en memoria cuatro arreglos bidimensionales de números enteros de 15x15. Los elementos de las matrices no serán cargados desde el teclado si no que cada una debe ser rellenado de la siguiente manera:

- Matriz *sumCoordenadas*: cada posición de la matriz debe ser rellenada con el valor resultante de la suma de la posición de fila y columna en las que está por cargarse. Mostrar la matriz resultante.
- Matriz *pRandom:* cada posición de la diagonal principal de la matriz debe ser rellenada con un número aleatorio comprendido en el intervalo [0, 9]. El resto de las posiciones deben rellenarse con el valor 0 (cero). Mostrar la matriz resultante.
- Matriz *sRandom:* cada posición de la diagonal secundaria (contra diagonal) de la matriz debe ser rellenada con un número aleatorio comprendido entre [0, 9]. El resto de las posiciones deben rellenarse con el valor 1 (uno). Mostrar la matriz resultante.

13. Desarrollar un programa para cargar un matriz llamada *valores* de *5x5* y luego permita:

- Calcular la suma de todos los valores cargados en la matriz *valores* y mostrarlos.
- Recorrer la matriz *valores* por fila o por columna buscando el mayor y el menor valor de la matriz y las posiciones en que se encuentran. Mostar los valores obtenidos.
- Calcular el promedio de cada fila y guardar los resultados en un vector llamado *promedioF.*
- Calcular el promedio de cada columna y guardar los resultados en un vector llamado *promedioC.*

14. El director de le Escuela de Ingeniería de la UNdeC requiere determinar con urgencia aquellos alumnos que cursan más de una carrera de ingeniería relacionada con sistemas. Dispone del número de DNI de todos los alumnos de cada carrera de sistemas de la escuela. Se solicita diseñar un programa que permita cargar los documentos de los alumnos de las carreras de Ing. en Sistemas e Ing. Mecatrónica en una matriz llamada *alIngenierias* de 2x20 elementos y luego determine aquellos alumnos comunes a ambas carreras. Estos alumnos se guardarán en un nuevo arreglo llamado *alumnosC* de 20 elementos y se mostrarán en pantalla.

15. Desarrollar un programa con el siguiente menú de opciones:

- **Opción 1 -** Cargar un arreglo unidimensional *V*.
  - Permitir cargar desde el teclado un arreglo de 10 números reales.
- **Opción 2 -** Cargar un arreglo bidimensional *A* por fila.
  - Cargar por fila un arreglo de números enteros de 10x15 elementos y mostrarlo.
- **Opción 3 -** Cargar un arreglo bidimensional *B* por columna.
  - Cargar por columna un arreglo de números enteros de 10x15 elementos y mostrarlo.
- **Opción 4 –** Sumar *A+B* en *C*
  - Generar la sumatoria de las matrices *A+B* en el arreglo *C*.
- **Opción 5 –** Restar *A-B* en *D*
  - Generar la resta de las matrices *A-B* en el arreglo *D*.
- **Opción 6 -** Salir.

Nota: El menú y cada una de sus opciones deberán implementarse usando funciones.

16. Diseñar un programa que invierta los elementos de un arreglo de 20 posiciones. De tal manera que el primer elemento pase a ser el último y el último el primero, el segundo pase a ser el penúltimo y el penúltimo el segundo, y así sucesivamente.

17. Diseñar un algoritmo que genere una matriz identidad de 35x35. Recuerden que una matriz identidad es aquella en la que todos sus elementos son 0 (cero), excepto los elementos de la diagonal principal que son 1 (uno).

```
Matriz identidad 35x35
```

```
10000000000000000000000000000000000
01000000000000000000000000000000000
00100000000000000000000000000000000
00010000000000000000000000000000000
00001000000000000000000000000000000
00000100000000000000000000000000000
00000010000000000000000000000000000
00000001000000000000000000000000000
00000000100000000000000000000000000
00000000010000000000000000000000000
00000000001000000000000000000000000
00000000000100000000000000000000000
00000000000010000000000000000000000
00000000000001000000000000000000000
00000000000000100000000000000000000
00000000000000010000000000000000000
00000000000000001000000000000000000
00000000000000000100000000000000000
00000000000000000010000000000000000
00000000000000000001000000000000000
00000000000000000000100000000000000
00000000000000000000010000000000000
00000000000000000000001000000000000
00000000000000000000000100000000000
00000000000000000000000010000000000
00000000000000000000000001000000000
00000000000000000000000000100000000
00000000000000000000000000010000000
00000000000000000000000000001000000
00000000000000000000000000000100000
00000000000000000000000000000010000
00000000000000000000000000000001000
00000000000000000000000000000000100
00000000000000000000000000000000010
00000000000000000000000000000000001
```

18. Realizar un algoritmo que obtenga la matriz *transpuesta* de una matriz *original* de tamaño *NxM*. El tamaño de ésta última matriz será definido por el programador al inicio.

19. Un teatro desea permitir reservar asientos de su sala a los espectadores que deseen asistir a alguna de las obras disponibles en cartelera. La sala del teatro tiene una dimensión de doce filas por treinta columnas de butacas resultando en una capacidad total de 360 espectadores.

Se solicita diseñar un programa que permita reservar un asiento en la sala por medio de una matriz llamada *butacasSala* de 12x30 elementos. La matriz debe rellenarse con ceros al inicio ya que asumiremos que cuando exista un número 0 (cero) en la posición, la butaca está libre para reserva.

Una vez cargada la matriz, se debe permitir realizar un número *n* de reservas solicitando en cada una la posición de la butaca deseada. La posición de cada asiento será la intersección de la fila y columna de donde se encuentra. Por medio de un cartel comunicar la situación del asiento seleccionado al espectador: en el caso que el asiento esté desocupado concretar la reserva reemplazando el valor de la posición por el valor uno y

mostrando un mensaje en pantalla tal como “Reserva exitosa”; en caso que el asiento este ocupado mostrar un mensaje en pantalla tal como “Butaca no disponible, seleccione otra” y volver a solicitar la posición de asiento deseada.

20. El profesor de la clase de Sistemas I tiene un grupo con una cantidad *n* de alumnos y ha decidido tomar cuatro (4) parciales durante el cursado de la asignatura. Para simplificar su trabajo solicita diseñar un programa que permita cargar las notas de los cuatro parciales para los *n* alumnos:

- El promedio de calificaciones de cada alumno.
- El rendimiento general por parcial
- El promedio general del grupo
- El número del alumno que tuvo el mayor promedio de calificación.

21. Diseñar un programa para cargar un arreglo de 10 elementos con números pares. Luego solicitar un número a buscar y determinar si existe o no en el arreglo. La búsqueda será secuencial.

22. Desarrollar un programa para cargar un arreglo de 6 elementos de tipo entero. Luego ordenar el mismo en forma ascendente utilizando los siguientes métodos:

- Método de la burbuja con recorrido de izquierda a derecha.
- Método de la burbuja con recorrido de derecha a izquierda.
- Método de la burbuja con test de comprobación.
- Método sacudida.

23. Diseñar un programa que permita rellenar un vector con 100 números enteros generados de manera aleatoria y ordenarlo de manera ascendente. Luego de ordenar el vector, solicitar al usuario un número a buscar e indicarle por medio de un mensaje si dicho número se encuentra en el arreglo o no. La búsqueda será binaria.

24. Desarrollar un programa que permita cargar dos matrices *M1* y *M2* con números enteros, para luego calcular la matriz producto *MP* resultante. Tener en cuenta que para poder realizar el producto el número de columnas de la primera matriz debe ser igual al número de filas de la segunda matriz por lo que deberá realizarse ésta validación al inicio. En caso de que los datos ingresados por el usuario no cumplan con esta condición, se debe volver a pedir que se ingresen los tamaños de las matrices hasta que se cumpla.

Nota: antes de generar la matriz producto inicializar la misma con ceros.

25. En memoria se encuentra almacenado un arreglo bidimensional llamado camposZ de *NxM* elementos que representan la cantidad promedio de lluvia mensual en ml registrada para un campo y una zona del mismo campo. Se considera que cada campo fue particionado para el estudio en distintas zonas. Se solicita diseñar un programa que por medio de un menú de opciones permita implementar las siguientes funcionalidades:

- **Opción 1 –** Configuración del estudio a realizar
  - Cantidad de campos (*N*) y cantidad de zonas por campo (*M*) para generar la matriz *camposZ*.
- **Opción 2 -** Cargar promedios de lluvia.

    - La carga puede ser manual o a través de la generación de valores aleatorios comprendidos en el intervalo [0, 500].
- **Opción 3 -** Promedio de lluvias por campo
    - Incluye todas las zonas del mismo campo. Los promedios de lluvia por campo deberán calcularse y cargarse en un nuevo vector llamado *promediosC*.
- **Opción 4 -** Promedio de lluvias por zona.
    - Incluye la misma zona de distintos campos. Los promedios de lluvia por zona deberán calcularse y cargarse en un nuevo vector llamado *promediosZ*.
- **Opción 5 -** Campo y zona con mayor promedio de lluvias.
    - Los mayores promedios de lluvia por campo deberán determinarse y cargarse en un nuevo vector llamado *promediosMay*.
- **Opción 6 –** Ver resultados.
    - Salida en pantalla de la matriz *camposZ* y de los vectores *promediosC, promediosZ y promediosMay.*
- ***Opción 7 –*** *Salir.*

Nota: implementar el menú y cada opción utilizando funciones.