Ejercicios Tema 1. Referencias, Direcciones y Punteros

1. Defina una función llamada *ejercicio1* que recibe una referencia a un flotante llamado r_f1 y no devuelve nada. La función considera que r_f1 es un ángulo de Euler (en grados) y debe obtener ese ángulo en radianes. Por ejemplo, si f1 = 90, después de ejecutar la función f1 debe ser 0.7853.
2. Defina una función llamada *ejercicio2* que recibe una variable de tipo *int* llamado *e1*, otra a tipo referencia *double* llamado *rd1*) y una variable de tipo *long* llamada *l1*; y no devuelve nada. La función *ejercicio2* actualiza el valor de *dd1* con el resultado de multiplicar *de1* por *l1* y dividir el resultado por el valor de *dd1*. Implemente un método principal que pruebe la funcionalidad de *ejercicio2.* Por ejemplo, defina una variable de tipo *int* llamada *a* con valor 10, una variable de tipo *double* llamada *b* con valor 8.1 y una variable de tipo *long* llamada *c* con valor 6. El valor de *b* (mostrado por pantalla) después de ejecutar la función *ejercicio2* será de 7.40741.
3. Defina una función llamada *ejercicio2* que recibe dos variables de tipo punteros (uno apunta a *int* llamado p*e1* y otro a tipo *double* llamado p*d1*) y una variable de tipo *long* llamada *l1*; y no devuelve nada. La función *ejercicio3* actualiza el valor de pd1 con el resultado de multiplicar p*e1* por *l1* y dividir el resultado por el valor de p*d1*. Implemente un método principal que pruebe la funcionalidad de *ejercicio3*. Por ejemplo, defina una variable de tipo *int* llamada *a* con valor 10, una variable de tipo *double* llamada *b* con valor 8.1 y una variable de tipo *long* llamada *c* con valor 6. El valor de *b* (mostrado por pantalla) después de ejecutar la función *ejercicio3* será de 7.40741.
4. Utilizando punteros, implemente una función que, dado un vector de elementos enteros, devuelva la suma de todos los elementos del vector, así como la media aritmética. Implemente además un método principal que pruebe la funcionalidad anterior.
5. Implemente la función *multiplicarVectores* que recibe dos vectores de enteros de dimensiones determinadas n y devuelve la multiplicación de ambos. Implemente además un método principal que pruebe la funcionalidad anterior.

   *int multiplicarVectores(int *v1, int *v2, int n)*

6. Considérese una lista formada por *NxPxQ* elementos de tipo *float*. La lista contendrá los números 1.0, 2.0, 3.0, etc. Se pide construir un programa que muestre las direcciones de memoria todos los elementos de la lista, empleando el operador &. N, P y Q son números enteros (por ejemplo 2, 3 y 4). Implemente además un método principal que pruebe la funcionalidad anterior.
7. Se dispone de dos matrices *MxN* de *float*. Se pide construir una función basada en punteros de float que permita calcula la suma de estas dos matrices. Implemente además un método principal que pruebe la funcionalidad anterior.
8. Se pide:
   a. Implemente una función llamada comparaDosMatrices que recibe dos punteros a matrices de doubles llamadas md1 y md2 de tamaño fil x col, siendo fil el número de filas y col el número de columnas. El objetivo de comparaDosMatrices es comparar ambas matrices, para ello devuelve: (a) la matriz de enteros int resultante de comparar uno a uno los elementos de las matrices md1 y md2, y (b) el número de elementos de md1 que son mayores que md2. La comparación es: si el elemento de la matriz md1 es mayor que el elemento de la matriz md2 en la misma posición, se establece el valor 1 en dicha posición de la matriz resultante,

si ambos elementos son iguales, el valor es 0, y si el elemento de md1 es menor que el de md2, entonces se establece el valor -1.
Por ejemplo, dadas dos matrices md1 y md2 de 2x2, el resultado será tal como se describe a continuación:

$$md1 = \begin{pmatrix} 12.2 & 2.4 \\ 62.36 & 44.2 \end{pmatrix}, md2 = \begin{pmatrix} 5.6 & 31.1 \\ 62.36 & 21.4 \end{pmatrix} \rightarrow mRes = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} y\ res = 2$$

El método comparaDosMatrices devolverá tanto mRes como res.

b. Implemente un método principal que pruebe la funcionalidad anterior con dos matrices de 3x1. Cree las matrices manera estática con valores fijos, NO SOLICITE POR TECLADO LOS DATOS DE LA MATRIZ, lo importante es probar el método comparaDosMatrices.

Ejercicios Tema 1. Gestión de la Memoria y Punteros

1. Dibujar evolución Zona Estática y Stack de las variables del siguiente fragmento de código:

```
#include <iostream>
using namespace std;
int funcion1(int, int);
int main() {
    int x = 7;
    x = funcion1(4, 8);
    cout << x << endl;
    return 0;
}
int funcion1(int arg1, int arg2) {
    int local1;
    int local2 = 5;
    local1 = arg1 + arg2 + local2;
    return (local1 + 3);
}
```

2. Dibujar evolución Zona Estática y Stack de las variables del siguiente fragmento de código:

```cpp
int g(int , int) ;
int f(int , int) ;
int main() {
    int x=3;
    x=f(5,6);
    cout << x << endl;
    return 0;
}

int g(int a, int b) {
    int c;
    cout << "Al entrar en g: a = " << a << " b = " << b << endl;
    a = a + b; /*cambia argumento a */
    c = a + 4;
    cout << "Antes de salir de g: a = " << a << " b = " << b << " c = " << c << endl;
    return (c);
}
int f(int a, int b) {
    int c;
    int d = 5;
    c = a + b + d;
    cout << "Antes de g: a = " << a << " b = " << b << " c = " << c << " d= " << d << endl;
    d = g(a, b + c); /*se copian los valores en el frame de g */
    a = a + d;
    b = b + a;
    cout << "Después de g: a = " << a << " b = " << b << " c = " << c << " d= " << d << endl;
    return (d + 2);
}
```

3. Dibujar evolución Zona Estática y Stack de las variables del siguiente fragmento de código:

```
int funcion1(int, int);
int main() {
    int x=3;
    x = funcion1(4, funcion1(2,3)) + x;
    cout << x << endl;
    return 0;
}
int funcion1(int arg1, int arg2) {
    int local1;
    int local2 = 5;
    local1 = arg1 + arg2 + local2;
    return (local1 + 3);
}
```

4. Dibujar evolución Zona Estática y Stack de las variables del siguiente fragmento de código:

```
int fibonacci(int);
int main()
{
    int  n = 4;
    int res = fibonacci(n);
    cout << "Fibonacci de " << n << " es " << res << endl;
}

int fibonacci(int n) {
    int res = 0;
    if (n -- 0)
        res = 0;
    else if (n == 1)
        res = 1;
    else
        res = fibonacci(n - 1) + fibonacci(n - 2);
    return res;
}
```

5. Dibujar evolución Zona Estática y Stack de las variables del siguiente fragmento de código:

```
void factorial(int* acum, int n);
int main()
{
    int * valAcumulado = new int;
    * valAcumulado = 0;
    factorial(valAcumulado, 6);
    cout << "Valor acumulado " << *valAcumulado << endl;
}

void factorial(int* acum, int n) {
    if (n > 0) {
        *acum = (*acum) * n;
        factorial(acum, n - 1);
    }
}
```

6. Dibujar evolución Zona Estática y Stack de las variables del siguiente fragmento de código:

```
int* direccion_menor(int * p, int size) {
   int *dir_menor = p;
   for (int i = 1; i < size; i++) {
       if (*p < *dir_menor) {
           dir_menor = p;
       }
       p++;
   }
   return dir_menor;
}
int main(){
    int vector[] = { 10, 5, 9, 4, 15 };
    int *elem_menor;
    elem_menor = direccion_menor(&vector[0], 5);
    cout << "En "<< elem_menor <<" el valor es " << *elem_menor;
}
```

7. Cree una función en C++ que permita controlar el aforo a un club VIP. La función recibe como parámetro la edad de la persona y retorna un bool igual a true si la persona ha ingresado, false si no. Las normas indican lo siguiente:
   - Si la persona tiene una edad igual o superior a 21 puede ingresar siempre y cuando exista sitio dentro del club.
   - El aforo se limita a 4 personas y al inicio no existen personas dentro.
   - Dentro de la función se debe utilizar una variable estática llamada aforo

   Realice una función principal que pruebe la funcionalidad del programa.
8. Cree una función en C++ que consista en el juego de adivinar un entero. La función recibe como parámetros: un bool que indica si es un juego nuevo o no y un número que corresponde a la predicción hecha por el usuario. La función retorna un bool igual a true si la persona ha adivinado de forma correcta el número. Las normas indican lo siguiente:
   - El número a adivinar se guarda dentro de una variable estática llamada numero_a_adivinar y se modifica exclusivamente cuando el bool juego_nuevo sea true.
   - Para modificar el bool se utiliza la instrucción rand()%10+1 (siendo rand de la include stdlib)
   - Si el numero predicho es igual al numero_a_adivinar se retorna un true.
   - De lo contrario, se imprime en pantalla si el numero predicho es mayor o menor que el numero_a_adivinar.

   Realice una función principal que pruebe la funcionalidad del programa.
9. Cree una variable y un puntero a dicha variable. Ambas variables deben estar en la memoria heap. No olvide de eliminar las variables.
10. Cree un arreglo de enteros de tamaño 500 en la memoria heap (int[500]). No olvide de eliminar las variables.
11. Cree una matriz de caracteres de tamaño 100x100 en la memoria heap (char[100] [100]). No olvide de eliminar las variables.