Universidad del Valle sede Tuluá
Facultad de Ingeniería
Ingeniería de Sistemas
Introducción a la Programación Orientada a Objetos
Doc: M.Sc. Adrian Lasso C – luis.lasso@correounivalle.edu.co

Taller Métodos que reciben y retornan valores en C++

1. Implementar un método que reciba una variable X de tipo long y retorne true si X es primo, o false en caso contrario.

```
bool esPrimo(long num){
  …….
 return bool;
}
```

2. Implementar un método que calcule y retorne un número aleatorio entre un rango de valores. El método debe recibir dos variables de tipo long, la primera corresponde al valor mínimo y la segunda al valor máximo del rango.

```
long aleatorioEntre(long minimo, long maximo){
  …….
 return numero;
}
```

3. Implementar un método que reciba una cantidad dada en segundos y los retorne en horas.

```
int deSegAHoras(int seg){
  …….
 return horas;
}
```

4. Implementar un método que reciba una cantidad dada en segundos y los retorne en minutos.

```
int deSegAMinutos(int seg){
  …….
 return minutos;
}
```

5. Implementar un método que reciba una cantidad dada en metros y los retorne en kilómetros.

```
long deMtsAKms(long mts){
  …….
 return kms;
}
```

6. Implementar un método que reciba una cantidad dada en metros y los retorne en pulgadas.

```
long deMtsAPulg(long mts){
  …….
 return pulg;
}
```

7. Implementar un método que reciba una cantidad dada en metros y los retorne en pies.
   long deMtsAPies(long mts){
     …….
    return pies;
   }

8. Implementar un método que reciba una cantidad dada en metros y retorne un arreglo de tamaño 3 en cuyas posiciones se almacena la cantidad en kilómetros, pulgadas y pies.
   long *deMtsATodo(long mts){
     …….
    return arreglo;
   }

9. Implementar un método que reciba un valor en pesos colombianos y retorne un arreglo de tamaño 3 en cuyas posiciones se almacena el valor en dólares EEUU, Euros y Libras esterlinas.
   long *dePesosATodo(long pesos){
     …….
    return arreglo;
   }

10. Implementar un método que reciba como argumento un arreglo A[ ] de tipo long. El método debe retornar el promedio (media aritmética) de los elementos de A[ ].
    Ayuda: https://www.vitutor.com/estadistica/descriptiva/a_10.html
    double promedio(long *A){
      …….
     return prom;
    }

11. Implementar un método que reciba como argumento un arreglo A[ ] de tipo long. El método debe retornar la mediana estadística de los elementos de A[ ].
    Ayuda: https://www.vitutor.com/estadistica/descriptiva/a_9.html
    double mediana(long *A){
      …….
     return medio;
    }

12. Implementar un método que reciba como argumentos un arreglo A[ ] y un valor B, ambos de tipo long. El método debe retornar un arreglo C[ ] que almacena A[ ] x B.
    Ayuda: en Física esto sería el producto de un escalar por un vector.
    https://www.fisicapractica.com/producto-escalar-vector.php
    long *escXVector(long *A, long B){
      …….
     return C;
    }

13. Implementar un método que reciba como argumentos un arreglo A[ ] y arreglo B[ ], ambos de tipo long de tamaño 3. El método debe retornar un arreglo C[ ] que almacena el producto cruz de A[ ] y B[ ]. Ayuda: en Física es también es conocido como el producto vectorial. https://www.geoan.com/analitica/vectores/producto_cruz.html

```
long producVectorial(long *A, long *B){
  …….
 return C;
}
```

14. Implementar un método que reciba como argumentos un arreglo A[ ] y un valor B, ambos de tipo long. El método debe retornar *true* si B existe en A[ ], o *false* en caso contrario.

```
bool existeValor(long *A, long B){
  …….
 return bool;
}
```

15. Implementar un método que reciba como argumentos un arreglo A[ ] y un valor B, ambos de tipo long. El método debe retornar la posición donde se encuentra la primera ocurrencia de B en A[ ], o -1 en caso de que B no exista en A[ ].

```
int buscarValor(long *A, long B){
  …….
 return pos;
}
```

16. Implementar un método que reciba como argumentos un arreglo A[ ] y un valor B, ambos de tipo long. El método debe retornar la cantidad de veces que aparece B en A[ ].

```
int frecuencia(long *A, long B){
  …….
 return frec;
}
```

17. Implementar un método que reciba como argumentos un arreglo A[ ] y arreglo B[ ], ambos de tipo long. El método debe retornar un arreglo C[ ] que almacena los elementos de A[ ] y los elementos de B[ ].

```
long *unir(long *A, long *B){
  …….
 return C;
}
```

18. Implementar un método que reciba como argumento un arreglo A[ ] de tamaño N, y retorne un arreglo B[ ] donde primero aparecen los números pares y después los números impares de A[ ]. A[ ] y B[ ] son de tipo long.

Por ejemplo: sea A[ ] = {17, **2**, **4**, 9, **8**, 27, **90**, 15};

Entonces, B[ ] = {**2**, **4**, **8**, **90**, 17, 9, 27, 15};

```
long *paresImpares(long *A){
  …….
 return B;
```

}

19. Implementar un método que reciba como argumento un arreglo A[ ] de tipo int, que almacena la representación en binario de un valor entero. El método debe retornar un arreglo B[ ] que almacena la operación NOT de A[ ]
    Ayuda: NOT retorna 1 cuando la entrada es 0. Y, retorna 0 cuando la entrada es 1.
    Por ejemplo: sea A[ ] = {1, 1, 0, 1, 0};
    Entonces B[ ] = {0, 0, 1, 0, 1}
    int *AND(int *A){
    …….
    return B;
    }

20. Implementar un método que reciba como argumentos un arreglo A[ ] y arreglo B[ ], ambos de tipo *int* y de tamaño N, que almacenan cada uno la representación en binario de un valor entero. El método debe retornar un arreglo C[ ] que almacena la operación A[ ] AND B[ ].
    Ayuda: AND retorna 1 cuando ambas entradas son 1. Cero en caso contrario.
    Por ejemplo: sea A[ ] = {1, **1**, 0, **1**, 1};
    Y, B[ ] = {0, **1**, 1, **1**, 0};
    Entonces C[ ] = {0, **1**, 0, **1**, 0}
    int *AND(int *A, int *B){
    …….
    return C;
    }

21. Implementar un método que reciba como argumentos un arreglo A[ ] y arreglo B[ ], ambos de tipo *int* y de tamaño N, que almacenan cada uno la representación en binario de un valor entero. El método debe retornar un arreglo C[ ] que almacena la operación A[ ] OR B[ ].
    Ayuda: OR retorna 1 cuando al menos una de las entradas es 1. Cero en caso contrario.

    Por ejemplo: sea A[ ] = {0, 1, 0, 0, 0};
    Y, B[ ] = {0, 1, 1, 1, 0};
    Entonces C[ ] = {**0**, **1**, **1**, **1**, 0}
    int *OR(int *A, int *B){
    …….
    return C;
    }

22. Implementar un método que reciba como argumentos un arreglo A[ ] y arreglo B[ ], ambos de tipo *int* y de tamaño N, que almacenan cada uno la representación en binario de un valor entero. El método debe retornar un arreglo C[ ] que almacena la operación A[ ] NAND B[ ].
    Ayuda: NAND retorna 0 cuando ambas entradas son 1. Uno en caso contrario.
    Por ejemplo: sea A[ ] = {1, 1, 0, 1, 0};
    Y, B[ ] = {0, 1, 1, 1, 0};
    Entonces C[ ] = {**1**, **0**, **1**, **0**, **1**}

```
int *NAND(int *A, int *B){
  …….
 return C;
}
```

23. Implementar un método que reciba como argumentos un arreglo A[ ] y arreglo B[ ], ambos de tipo *int* y de tamaño N, que almacenan cada uno la representación en binario de un valor entero. El método debe retornar un arreglo C[ ] que almacena la operación A[ ] NOR B[ ].
    Ayuda: NOR retorna 1 cuando ambas entradas son 0. Cero en caso contrario.
    Por ejemplo: sea A[ ] = {1, 1, 0, 1, 0};
    Y, B[ ] = {0, 1, 1, 1, 0};
    Entonces C[ ] = {**0**, **0**, **0**, **0**, **1**}

```
int *NOR(int *A, int *B){
  …….
 return C;
}
```

24. Implementar un método que reciba como argumento un valor X de tipo *int*. El método debe retornar el histograma (formado con asteriscos) de X como una cadena de caracteres.
    Por ejemplo: sea X = 7;
    Entonces el método retorna = "*******"

```
string histograma(int X){
  …….
 return cadena;
}
```

25. Implementar un método que reciba como argumento un arreglo A[ ] tipo *int*. El método debe retornar un arreglo C[ ] de tipo *string* que almacena el histograma (formado con asteriscos) de cada uno de los elementos de A[ ]
    Por ejemplo: sea A[ ] = {2, 5, 3, 7, 10};
    Entonces C[ ] = {"**", "*****", "***", "*******", "**********"};

```
string *histogramas(int *A){
  …….
 return C;
}
```

26. Implementar un método que reciba como argumentos un matriz A[ ][ ] tipo *int* de tamaño N x M, y un valor X. El método debe retornar un arreglo B[ ] de tipo *int* de tamaño N que almacena en cada posición la frecuencia de X en cada fila de A[ ][ ].
    Por ejemplo;

Sea A=

2	**5**	10	8	9
1	12	**5**	9	7
5	**5**	4	12	**5**
12	10	3	**5**	9
3	2	6	9	6

Sea X = 5, entonces B =

1
1
3
1
0

```
int *frecuencia(int *A, int X){
   …….
  return B;
}
```