|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **NOMBRE DE LA PRÁCTICA** | **Prácticas de C ++** | | | | **No.** | | **1** |
| **ASIGNATURA:** | **Métodos númericos** | **CARRERA:** | **INGENIERIA EN SISTEMAS COMPUTACIONALES** | **PLAN:** | | **ISIC-2010-224** | |

**JOSÉ MARTÍNEZ HERNÁNDEZ 3042**

**I. COMPETENCIA(S) ESPECÍFICA(S): Aprender c++**

**II. MATERIAL EMPLEADO:**

* Laptop con Linix Lite
* Editor de texto Visual Studio Code
* Terminal

**III. DESARROLLO**

**Lámina 7 Entrada y Salida de datos**

**1.¿Cuál es la diferencia entre puts() y printf()?** Prinf acepta tanto cadenas como argumentos y ocupa un especificador de condición sin salto de linea automaticos, y el puts solo acepta cadena de caracteres y da un salto de linea automaticamente

**2. ¿Qué archivo de encabezado debe ser incluido cuando se usa printf?**

#include<stdio.h>

**3. ¿Qué especificador de conversión debe ser usado para imprimir lo siguiente?**

**a) Una cadena** = %s

**b) Un entero con signo** = %d

**c) Un número decimal de punto flotante =** %f

**d) Un carácter =** %c

**Encuentra el error**

**4. puts (“El precio es %d ”, precio );** Puts no acepta variables numéricas

**5. printf (“Esta es una cadena”)** Le hace falta el punto y coma

**6. scanf (“%f %d %f”, x, y, z);**  Le hace falta el & para saber donde lo guardara en memoria y las variables no han sido declaradas con anterioridad

**7. printf( %d, x );** Le hacen faltan las comillas

**8. scnaf (“%d”, &x)** Esta mal escrito el scanf

**Lámina 8 if**

**Ejercicio número 1: Números pares**

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre número y valor 34



* Condición si la división de número entre 2 da como residuo 0 entonces:



* Imprimir el mensaje en consola de el número es par

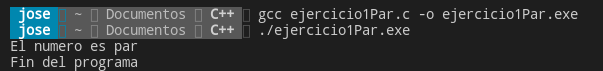


* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

Salvamos, compilamos y ejecutamos el programa:



El programa imprimió que el número era par porque entro a la condición ya que la sentencia era verdadera

Escribe un programa que reciba de teclado la calificación del examen de un alumno e imprima el siguiente texto únicamente si la calificación fue aprobatoria.

Mensaje: "¡Felicidades aprobaste el curso!"

La calificación mínima aprobatoria es: 8

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre calificación y valor 0



* Imprimir el mensaje que ingrese su calificación



* El usuario ingresara el valor de calificación por teclado:



* Condición si la calificación es igual o mayor que 8 entonces:



* Imprimir el mensaje:

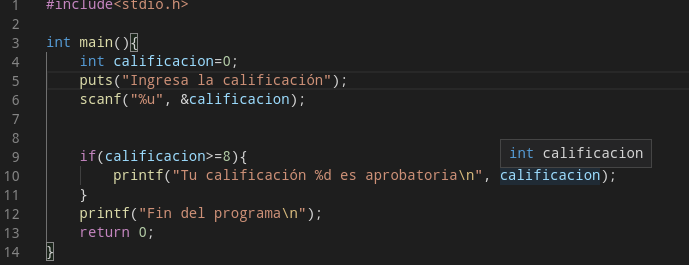


* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

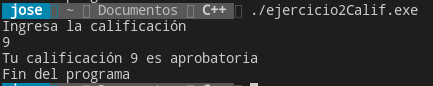
Código completo:



Al correr el programa e ingresar un dato menor que 8 solo mostrara que el programa ha terminado



En cambio si ingresamos 8 o mayor entonces



**If anidado**

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre numero y valor 55



* Condición si el numero es menor a 100 entonces:



* Si entonces imprimir el mensaje:



* Si entonces condición si el número es mayor que 50



* Si entonces imprimir el mensaje

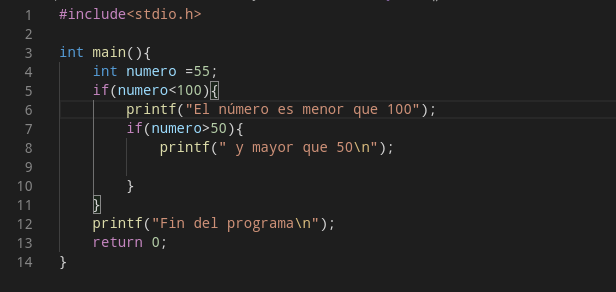


* Imprimir en consola el programa ha finalizado

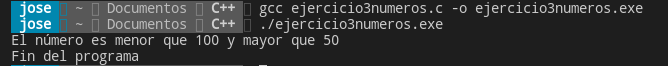


* Retornar 0 de nuestro método main

Código completo:



Al ejecutar el programa va a entrar a las condiciones dando como resultado:



* Incluimos las bibliotecas de entrada y salida de datos



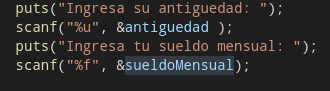
* Método principal



* Declaración de la variable int con nombre antigüedad,valor 0 y la variable tipo float con nombre sueldoMensual y valor 0



* Pedir las dos variables anteriores



* Condición si la antigüedad es mayor a o igual a 5:



* Si entonces condición si el sueldoMensual\*100/10 es mayor a 1000



* Si entonces imprimir el mensaje

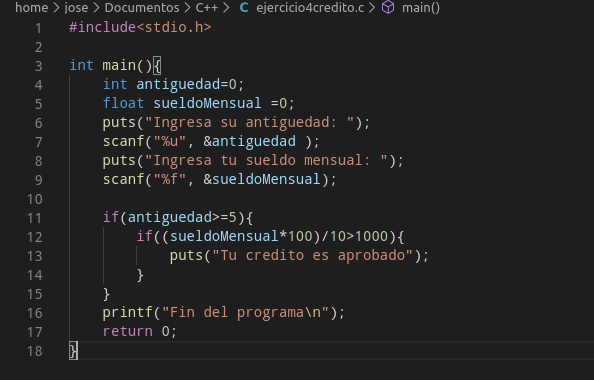


* Imprimir en consola el programa ha finalizado



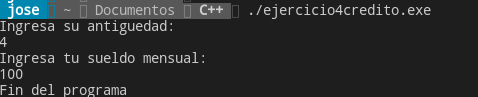
* Retornar 0 de nuestro método main

Código completo:

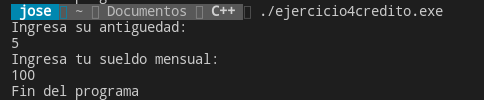


* Al compilar y ejecutar el código probemos las 4 combinaciones, el programa solo se ejecutara cuando las dos sentencias sean correctas

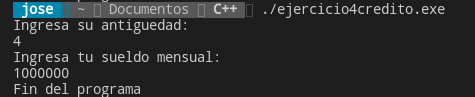
Antigüedad incorrecta y antigüedad incorrecta



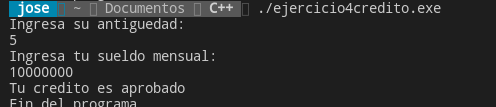
Antigüedad correcta pero sueldo incorrecto



Antigüedad incorrecta pero sueldo correcto



Antigüedad correcta y sueldo correcto



**Uso del else**

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre numero y valor 36



* Condición si el numero es menor a 50 entonces:



* Si entonces imprimir el mensaje:



* Si no entonces se imprime el mensaje

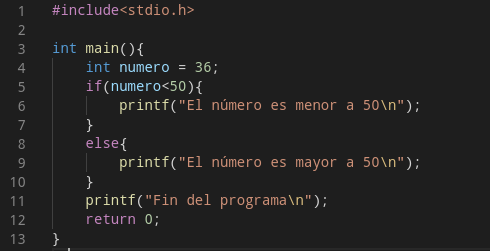


* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

Código completo:



* Al compilar y ejecutar el programa va dar como resultado el número es menor que 50 ya que el número tiene como valor 36:



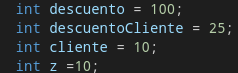
* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre descuento y valor 100, la variable int con nombre descuentoCliente y valor 25, la variable int con nombre cliente y valor 10, y por ultimo la variable int con nombre z y valor 10



* Condición si z es mayor o igual a cliente se ejecutara descuentoCliente si no sera descuento:



* Si entonces imprimir el mensaje:



* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

Al compilar y ejecutar el programa da como resultado

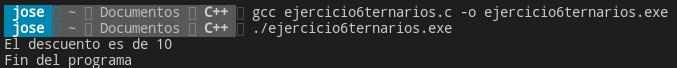


Lámina 9 While

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre contador y valor 0



* El ciclo while se repetirá hasta que el contador sea menor que 3:



* Cuando sea cierta la condición de while se mostrara el siguiente mensaje:



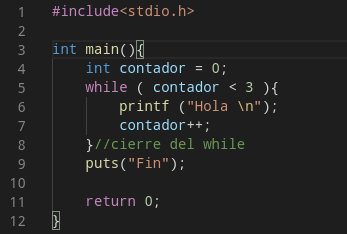
* Aqui va el contador que es una de las partes indispensables porque si no se ejecutaria el ciclo hasta el infinito



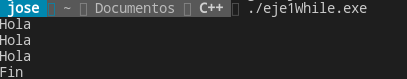
* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

Código completo

Al compilar y ejecutar el programas nos damos cuenta que el while se repitio 3 veces ya que fue del 0, 1, 2 hasta detenerse porque el siguiente número seria 3 y ya no seria cierta la condición



Ejemplo Número 2

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre contador y valor 1



* El ciclo while se repetirá hasta que el contador sea menor igual que 1000:



* Cuando sea cierta la condición de while se mostrara el siguiente mensaje:



* Aquí va el contador que es una de las partes indispensables porque si no se ejecutaría el ciclo hasta el infinito

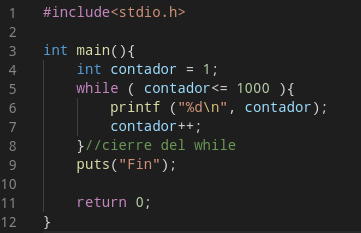


* Imprimir en consola el programa ha finalizado

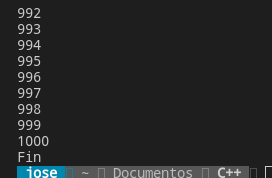


* Retornar 0 de nuestro método main

Código completo



Al compilar y ejecutar el código nos daremos cuenta que se imprimira del 1 al 1000 con salto de linea



Ejemplo Número 2

Imprimir los cuadrados y los cubos de los primeros

quinientos números naturales.

El cuadrado de un número se define como: x 2 = x\*x

El cubo de un número se define como: x 3 = x\*x\*x

La salida debe ser la siguiente:

1, 1, 1

2, 4, 8

3, 9, 27

4, 16, 64

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre contador y valor 1



* El ciclo while se repetirá hasta que el contador sea menor igual que 500:



* Cuando sea cierta la condición de while se mostrara el siguiente mensaje:



* Aquí va el contador que es una de las partes indispensables porque si no se ejecutaría el ciclo hasta el infinito



* Imprimir en consola el programa ha finalizado

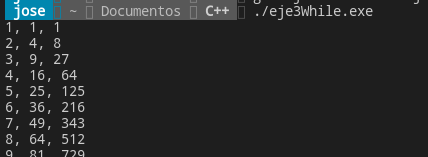


* Retornar 0 de nuestro método main

Código completo:



Al complar y ejecutar el código cumple de manera correcta el planteamiento del problema ya que da el número, el cuadrado del número y el cubo del mismo número luego da un salto de linea



Ejercicio Número 3:

Imprimir todos los números divisibles entre 3

mayores a 0 y menores a mil.

Ejemplo:

3, 6, 9, 12, 15, 18, 21, .... 999

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre contador y valor 1



* El ciclo while se repetirá hasta que el contador sea menor igual que 333, este número es porque al multiplicar 3 \* 333 da como resultado el 999 que es uno menor a 1000 y pues es correcto :



* Cuando sea cierta la condición de while se mostrara el siguiente mensaje:



* Aquí va el contador que es una de las partes indispensables porque si no se ejecutaría el ciclo hasta el infinito

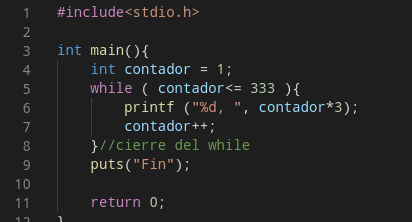


* Imprimir en consola el programa ha finalizado

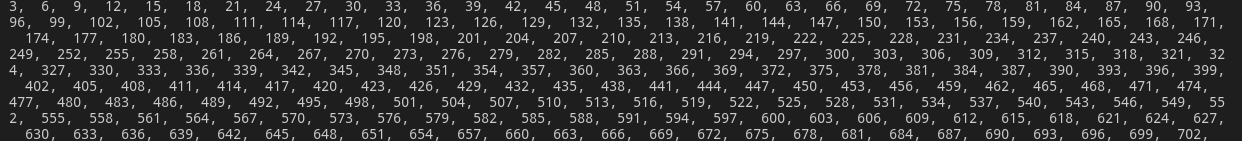


* Retornar 0 de nuestro método main

Código completo:



Al compilar y ejecutar el programa da como resultado lo pedido por el ejercicio



Ejercicio Número 4:

Imprimir todos los números que son divisibles entre 2

y entre 7, mayores a 0 y menores a mil.

Ejemplo:

14, 28, 42, 56, …

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre contador y valor 1



* El ciclo while se repetirá hasta que el contador sea menor igual que 1000 :



* Condición si el número divido entre 7 y 2 da como residuo 0



* Se mostrara el siguiente mensaje:



* Aquí va el contador que es una de las partes indispensables porque si no se ejecutaría el ciclo hasta el infinito



* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

Código completo:



Al compilar y ejecutar el programa da como resultado lo pedido por el ejercicio



Ejercicio Número 5:

Escribir todos los enteros positivos menores que 100

omitiendo aquellos que son divisibles por 7.

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre contador y valor 1



* El ciclo while se repetirá hasta que el contador sea menor igual que 100 :



* Condición si el número divido entre 7 da como residuo distinto a 0



* Se mostrara el siguiente mensaje:



* Aquí va el contador que es una de las partes indispensables porque si no se ejecutaría el ciclo hasta el infinito



* Imprimir en consola el programa ha finalizado

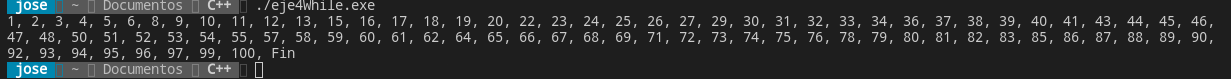


* Retornar 0 de nuestro método main

Código completo:



Al compilar y ejecutar el programa da como resultado lo pedido por el ejercicio



Do – While

La principal diferencia entre el do – while y el while es que el primero se ejecutara mínimo una vez

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración de la variable int con nombre i y valor 0



* Iniciamos con el do



* Cada que se repetirá se mostrara el siguiente mensaje



* La sentencia que condiciona el do es que la variable i sea menor a 3

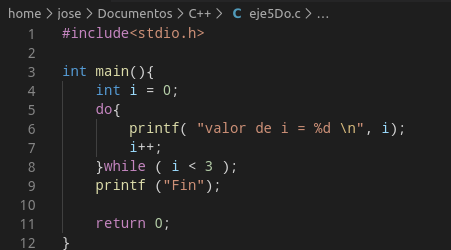


* Imprimir en consola el programa ha finalizado



* Retornar 0 de nuestro método main

Código completo



Al compilar y ejecutar el programa nos damos cuenta que el ciclo se se ejecuta 3 veces ya que una es la de cajón y dos extras hasta donde ya no cumple a condición estipulada

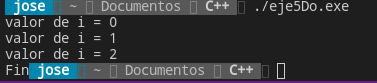


Lámina 10 Arreglos

Ejercicio número 1:

Crear un programa que declare un arreglo llamado "vector" de 10 posiciones.

Asignar el valor de 10 a cada posición del arreglo.

Mostrar todas las posiciones del arreglo.

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración del arreglo int con nombre arreglo y 10 dimensiones



* Declaración de la varible int con nombre contador y valor a 0



* While con condición que cubre de 0 a 9 puesto que son todas las posiciones del arreglo



* Mientras se ejecuta el while dar valor a las posiciones del arreglo e imprimirlas



* Aumentar en uno el contador para recorrer cada posición

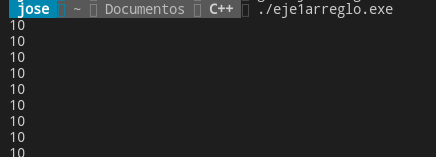


* Retornar 0 de nuestro método main

Código completo:



Al compilar y ejecutar el programas cumple con las características de lo pedido en el ejercicio



Ejercicio 2:

Crear un arreglo de 100 posiciones.

Llenar el arreglo con la tabla del 2.

Mostrar el arreglo en pantalla.

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración del arreglo int con nombre arreglo y 100 dimensiones



* Declaración de la varible int con nombre contador y valor a 0



* While con condición que cubre de 0 a 100 puesto que son todas las posiciones del arreglo



* Mientras se ejecuta el while dar valor a las posiciones del arreglo e imprimirlas, el valor del arreglo será el contador +1 por 2 para que abarque de 2 a 200



* Aumentar en uno el contador para recorrer cada posición

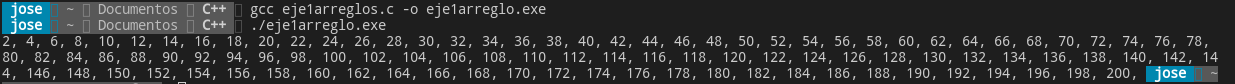


* Retornar 0 de nuestro método main

Código completo:



Al compilar y ejecutar el programas cumple con las características de lo pedido en el ejercicio numeración de 2 a 200



Ejercicio 3:

Crear un arreglo de 100 posiciones.

Llenar el orden inverso al indice.

Mostrar el arreglo en pantalla.

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración del arreglo int con nombre arreglo y 100 dimensiones



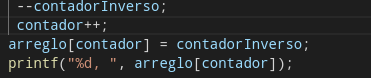
* Declaración de la varible int con nombre contador y valor a 0, y la variable int con nombre contadorInverso igual a 100



* While con condición que cubre de 0 a 100 puesto que son todas las posiciones del arreglo

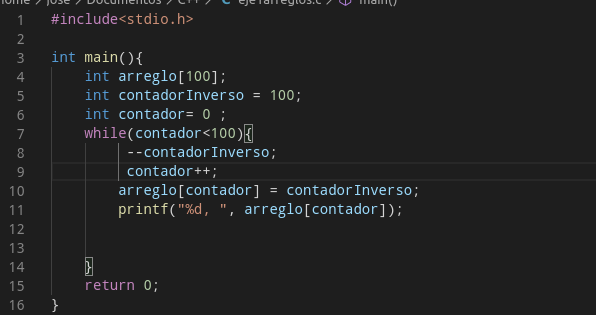


* Mientras se ejecuta el while dar valor a las posiciones del arreglo e imprimirlas, el valor del contador hira aumentado porque es el encargado de la las posiciones en cambio el valor del contadorInverso se ira disminuyendo ya que será el valor de cada posición

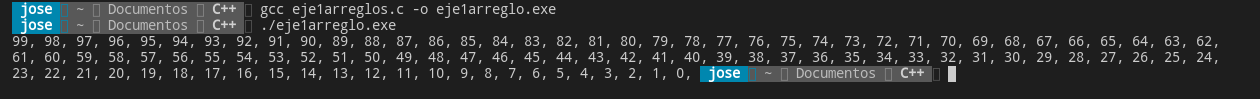


* Retornar 0 de nuestro método main

Código completo:



Al compilar y ejecutar el programas cumple con las características de lo pedido en el ejercicio numeración de 99 a 0



Ejercicio Número 4:

Dados

A = [3, 5, 6, 8, 4, 7, 8, 5, 3, 1]

B = [3, 4, 6, 8, 9, 1, 2, 3, 0, 9]

Realizar las siguientes operaciones

A[3] mod (B[2]/2)

B[A[1]] – A[9]

A[0] + A[1+2]

A[5] + B[5]

(A[3]/B[2])/2

* Incluimos las bibliotecas de entrada y salida de datos



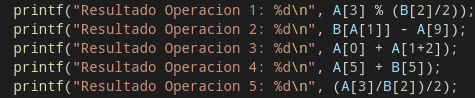
* Método principal



* Declaración del arreglo int con nombre a y b con dimensiones ambos y en la misma linea dar valores a las posiciones

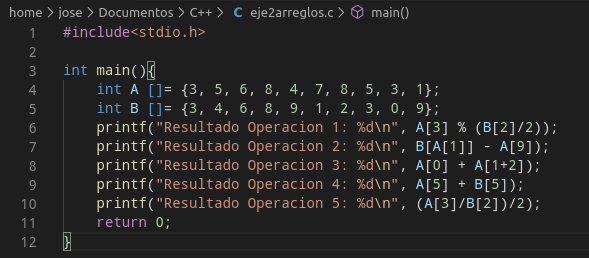


* Lo siguiente es imprimir las operaciones planteadas



* Retornar 0

Código completo



Al compilar y ejecutar el programa nos da los siguientes valores



Ejercicio Número 5:

Crea un arreglo de 20 posiciones.

Asígnale a cada elemento un valor.

Calcula el promedio de todos los elementos.

Calcula la multiplicación de todos los elementos.

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declaración del arreglo int con nombre A con dimensión 20 y en la misma linea dar valores a las posiciones



* Declaración de 3 variables de tipo int: contador, multiplicación y promedios, respectivamente para aumentar las posiciones y las otras dos van a almacenar la multiplicación y promedio de todos los elementos



* Lo siguiente es iniciar un while para que recorra posición por posición



* Después la variable multiplicación almacenara la multiplicación de cada posición al igual que promedios guardara cada suma de cada posición



* Lo último del while es aumentar el contador



* Despues la variable promedios se dividira entre 20 para poder obtener el promedio

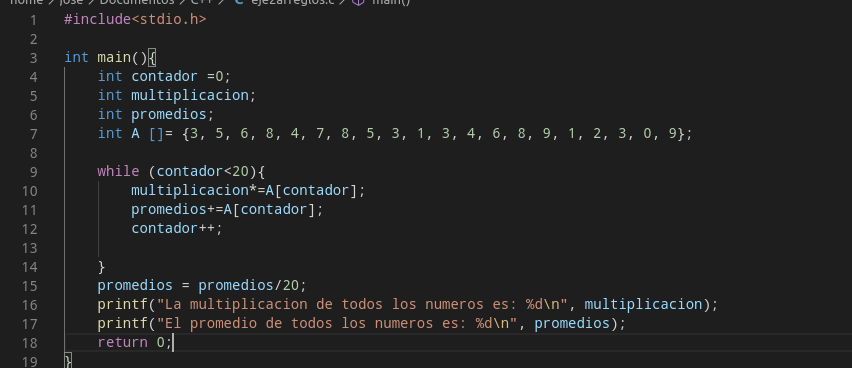


* Lo último es mostrar los resultados por consola



* Retornar 0

Código completo



* Cuando compilamos y ejecutamos nos da como resultado lo siguiente:

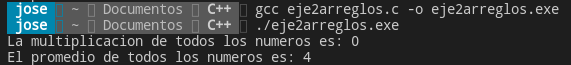


Lámina 11 For

Ejemplo 1:Mostrar la serie: 2, 3, 4, 6, 6, 9, 8, 12, 10,15

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declarar e inicializar la variable longitudSerie:



* Contador de el ciclo for: una variable int



* El ciclo for se ejecutara 5 veces

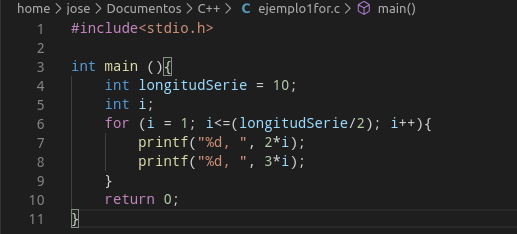


* La impresión de esta serie porque el patrón a seguir es multiplicar los número del 1 a 5 por 2 y 3



* Retornar 0

Código completo:



* Al compilar y ejecutar el programa da como resultado la serie pedida en el ejemplo



Ejercicio 1 :

Escribe un programa que reciba un número N del usuario y

haga la suma de todos los números desde 1 hasta N.

Ej.

1+2+3+4+5 = 15

* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



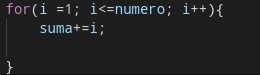
* Declarar 3 variables: contador llamado i; número a ingresar por el usuario llamada la variable número y el almacenador de la suma



* Pedir al usuario por teclado el numero y guardarlo ej numero



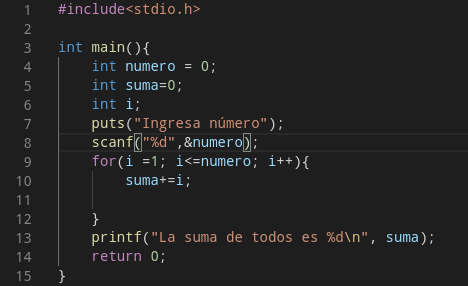
* Para poder ver todos lo números antes hay que hacer un for que repita desde 1 hasta llegar el número ingresado, luego sumarlos todos



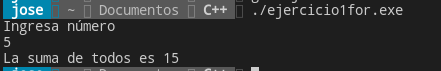
* Al final mostrar la suma en pantalla y hacer un salto de linea y retornar 0



Código completo

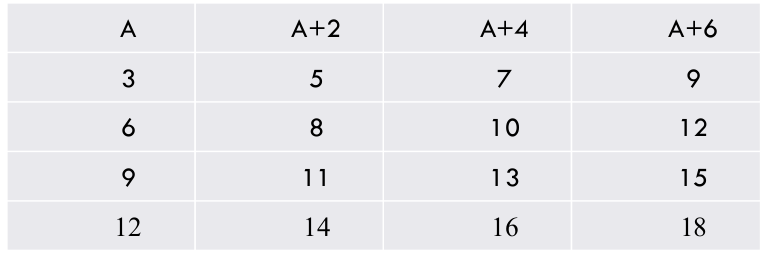


* Al compilar y ejecutar el código e ingresar un 5 da como resultado 15 y pues es correcto



Ejercicio 2:

Escriba un programa en C que utilice un ciclo para producir la siguiente tabla de valores



* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



* Declarar variables int llamado i que será el contador;



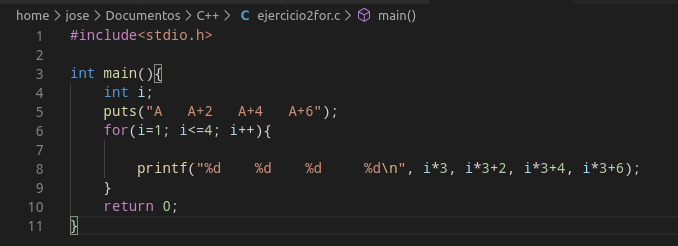
* Imprimir la fila de arriba



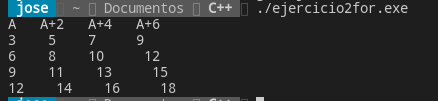
* Para poder imprimirlo como esta en la tabla primero se multiplicara \*3 y luego se sumara lo que dice su correspondiente columna



Código completo



Al compilar y ejecutar el programa veremos que salio la tabla pero un poco desalineada



For anidado:

Ejercicio 2:

Escribe un programa que reciba un número entero N del usuario e imprima una escalinata de N pisos de asteriscos



* Incluimos las bibliotecas de entrada y salida de datos



* Método principal



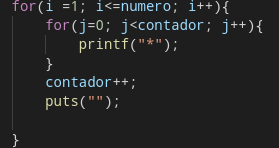
* Declarar variables int llamado i que será el contador, j para el otro contador y numero que almacenara el valor de lo ingresado por teclado;



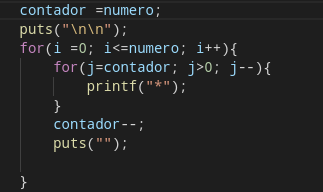
* Lo siguiente será pedirle el número al usuario



* El problema se ejecura de la siguiente manera que es que el primer ciclo se utilizara para repetir de 1 al número ingresado por el usuario, el segundo for se ejecutara del 1 al tamaño del contador que inicializara en 1 y cada que se acabe el segundo ciclo se aumentara uno al contador, en el segundo ciclo se imprimira un asterico pero sin salto de linea, al concluir el segundo ciclo se dara un salto de linea



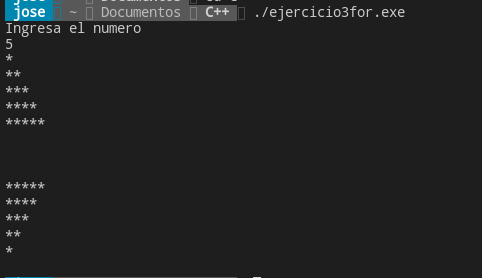
* Para el segundo ejercicio es totalmente igual solo que el contador tomara el valor del número, el segundo ciclo cambia ahora el contador ya esta lleno por asi decirlo y hay que vacearlo con un decremento de uno en uno para llegar a 1 y ahi dar el salto de linea



Código completo:

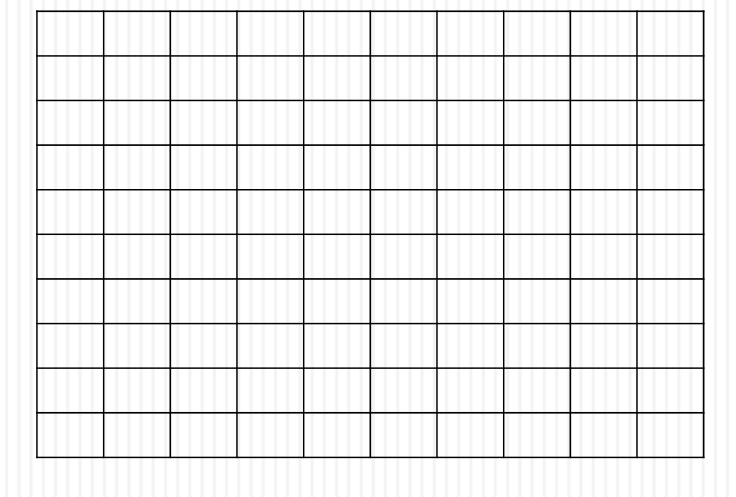


Al compilar y ejecutar el programa e ingresar un 5 ene teclado el resultado es correcto



Ejercicio 3:

Haz un programa que muestre las tablas de multiplicar como se muestra a continuación.



* Incluimos las bibliotecas de entrada y salida de datos



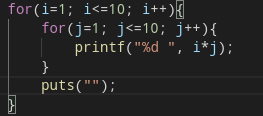
* Método principal



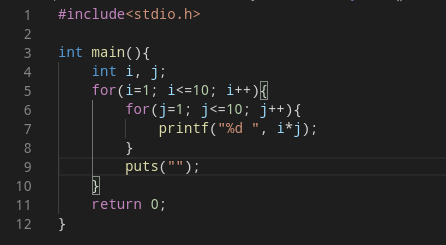
* Declarar variables int llamado i que será el contador j



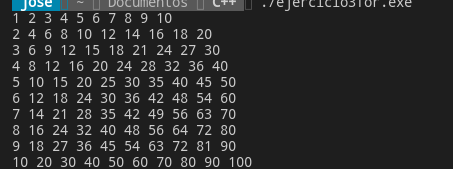
* Son dos ciclos 1 principal y otro el anidado que tienen la misma condición se pararan hasta llegar a 10, cuando se ejecute el anidado se mostrara la impresión de i \* j y cuando acabe se hara el salto de linea



Código completo:



Al compilar y ejecutar el programa es lo que pide el ejercicio aunque desalineado



IV. Conclusión:

Al acabar todos los ejemplos y ejercicios propuestos puedo llegar a la conclusión que las láminas están muy ilustrativas y entendibles lo cual facilito en la compresión de estos temas, los temas abordados en las láminas son los pilares en la programación ya que eran el if/else, la entrada y salida de datos y las sentencias repetitivas que son el while, do-while y for. Estos ejercicios me han ayudado de gran manera para mejorar mi lógica de programador.