

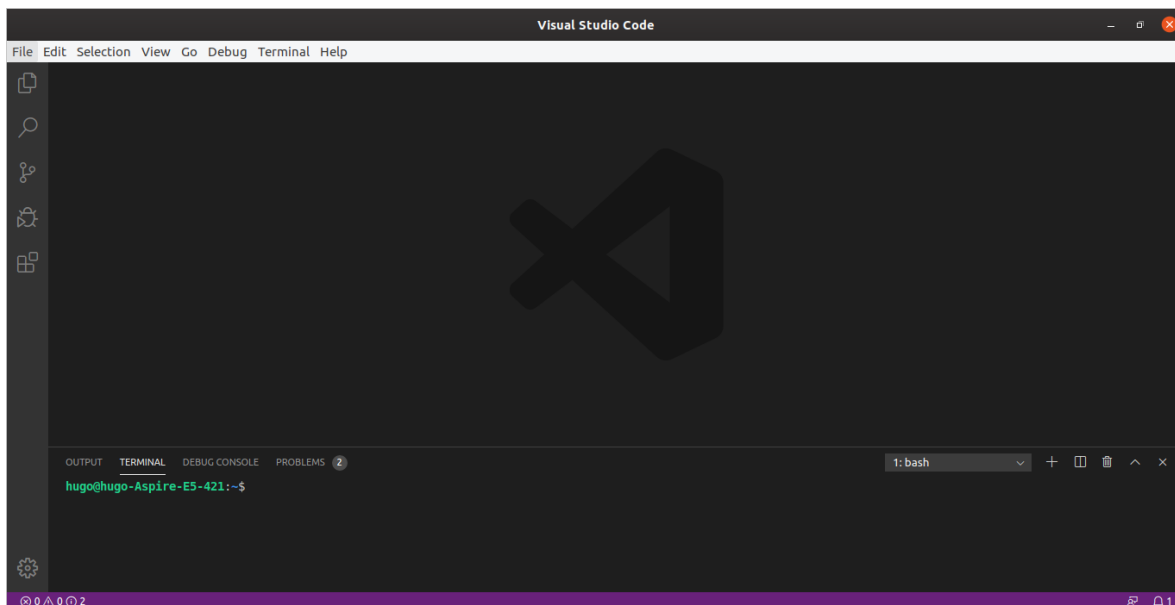
Nombre de la práctica	PROGRAMAS C			No.	1
Asignatura:	MÉTODOS NUMÉRICOS	Carrera:	ISIC	Duración de la práctica (Hrs)	4

valdez robles victor hugo isic 321

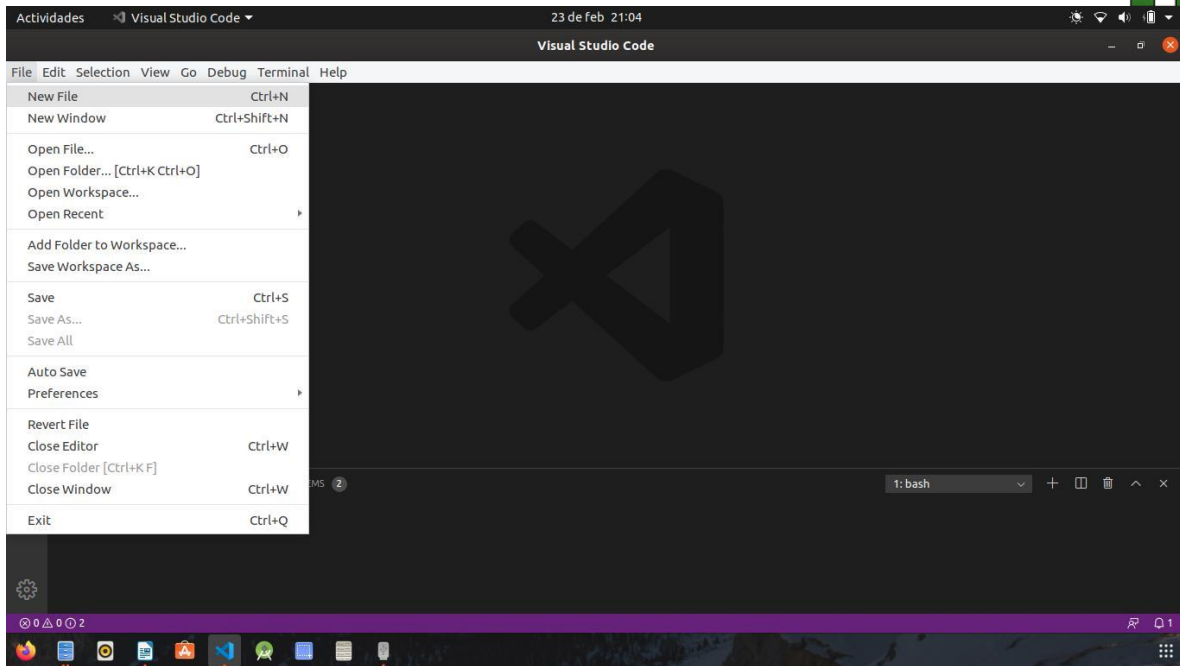
II. Lugar de realización de la práctica (~~laboratorio, taller, aula u otro~~):
TALLER

III. Material empleado:
Dev c++

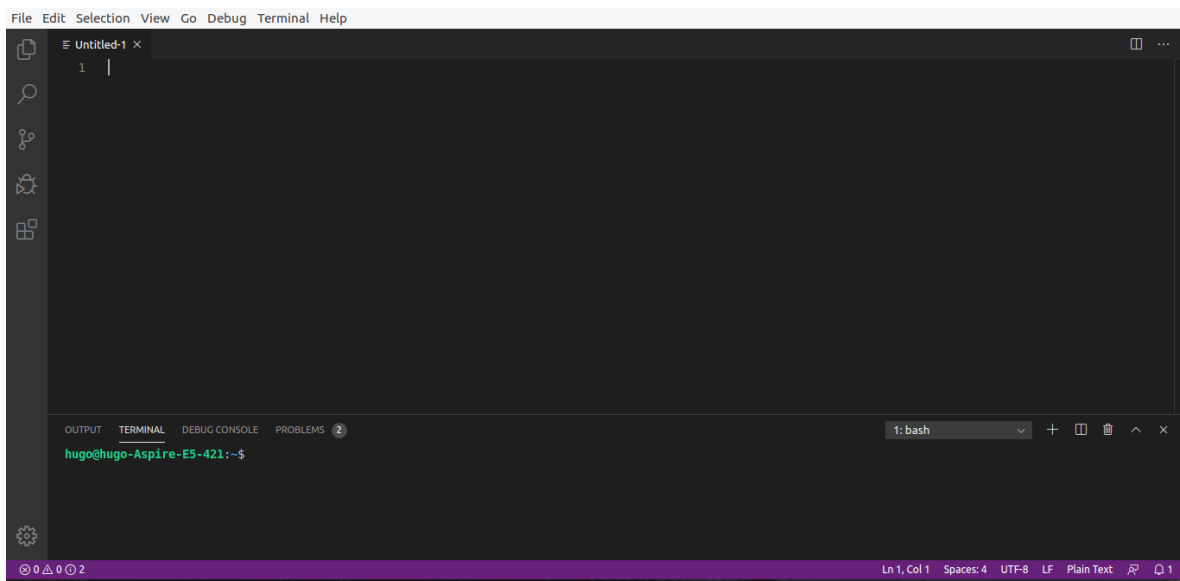
IV. Desarrollo de la práctica:
se abre la aplicación de visual studio code



Para crear un proyecto, nos dirigimos a la parte de file y señecionamos “newfile ” Al dar clic nos dara una pantalla para desarrollar



En esta parte se muestra la platilla para el código, donde los proyectos se deben de guardar con punto c, para poder ejecutar y compilar en caso de errores poder corregir para que realice su función de manera correcta.



Para empezar con nuestros programas se van a incluir los estándares de entrada y salida, las bibliotecas, el método principal donde estará nuestro código de forma estructurada.

```
home > hugo > Escritorio > programasC > C Aa.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
```

Para todos los codigos siguientes se va a crear con un metodo principal para ejecutar los codigos que se hace de la siguiente manera

```
home > hugo > Escritorio > programasC > C Aa.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5
6
7      system("pause");
8
9      return 0;
10 }
```

en todos los codigos siguientes , en la parte superior se mostraran la estructura de los mismos y en la inferior la compilación y ejecucion

El primer codigo a desarrollar es Hola Mundo que es el siguiente



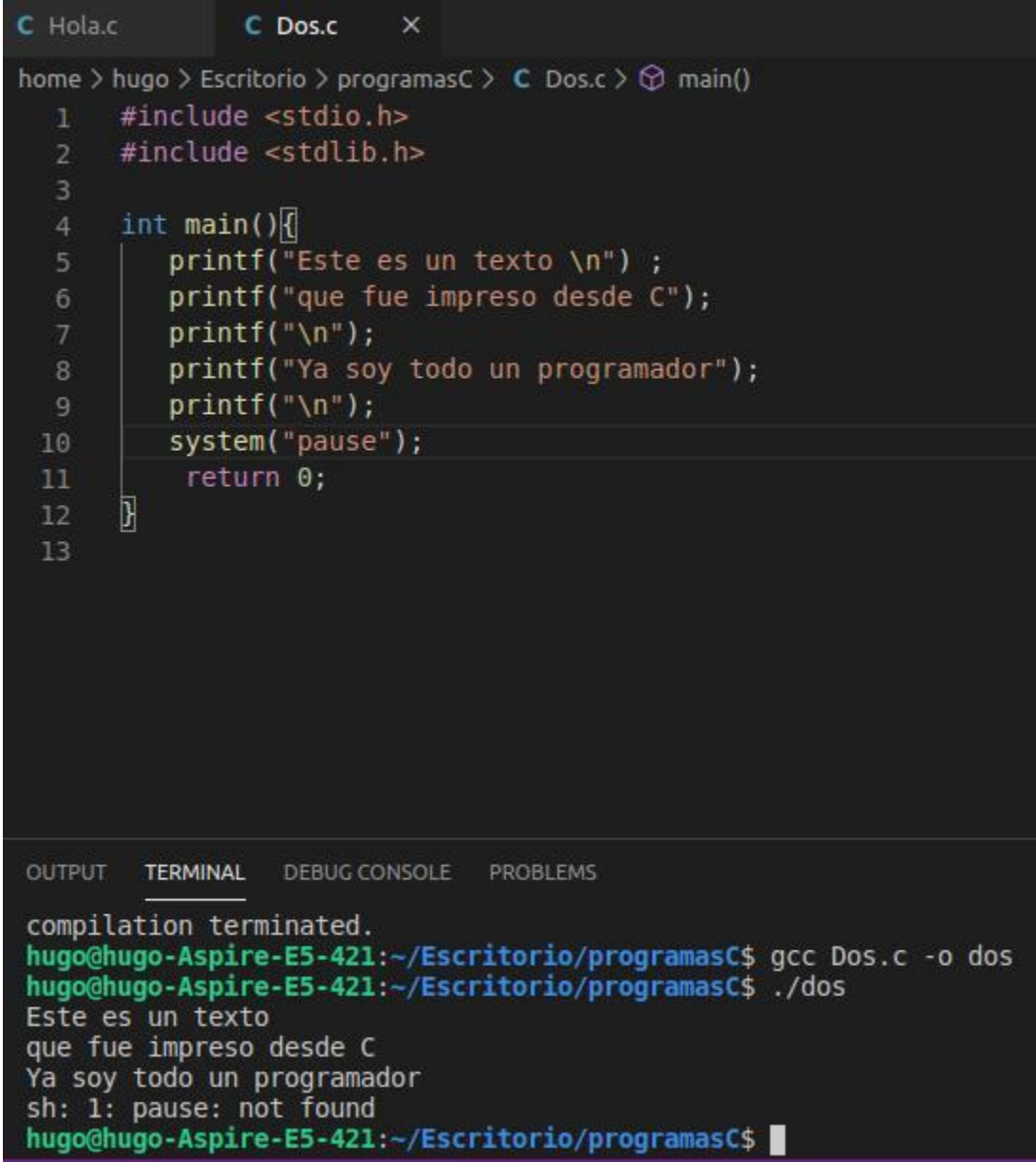
```
C Operadores0.c  C Hola.c  X
home > hugo > Escritorio > programasC > C Hola.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      printf("hola mundo \n");
6
7      //system("pause");
8
9      return 0;
10 }
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
ingresa calificacion
7
reprobado
sh: 1: Pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Hola.c -o hola
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./hola
hola mundo
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

Mostraremos una secuencia donde se imprimirá un mensaje en la que (\n) nos funcionara como un salto de línea esto quiere decir que el otro mensaje se pasara al siguiente renglón. Para cerrar el programa se pone un system con un pause, terminando con 0.

Entramos a símbolo de sistema, compilamos y ejecutamos, nos muestra en pantalla lo que tenemos en nuestro código.



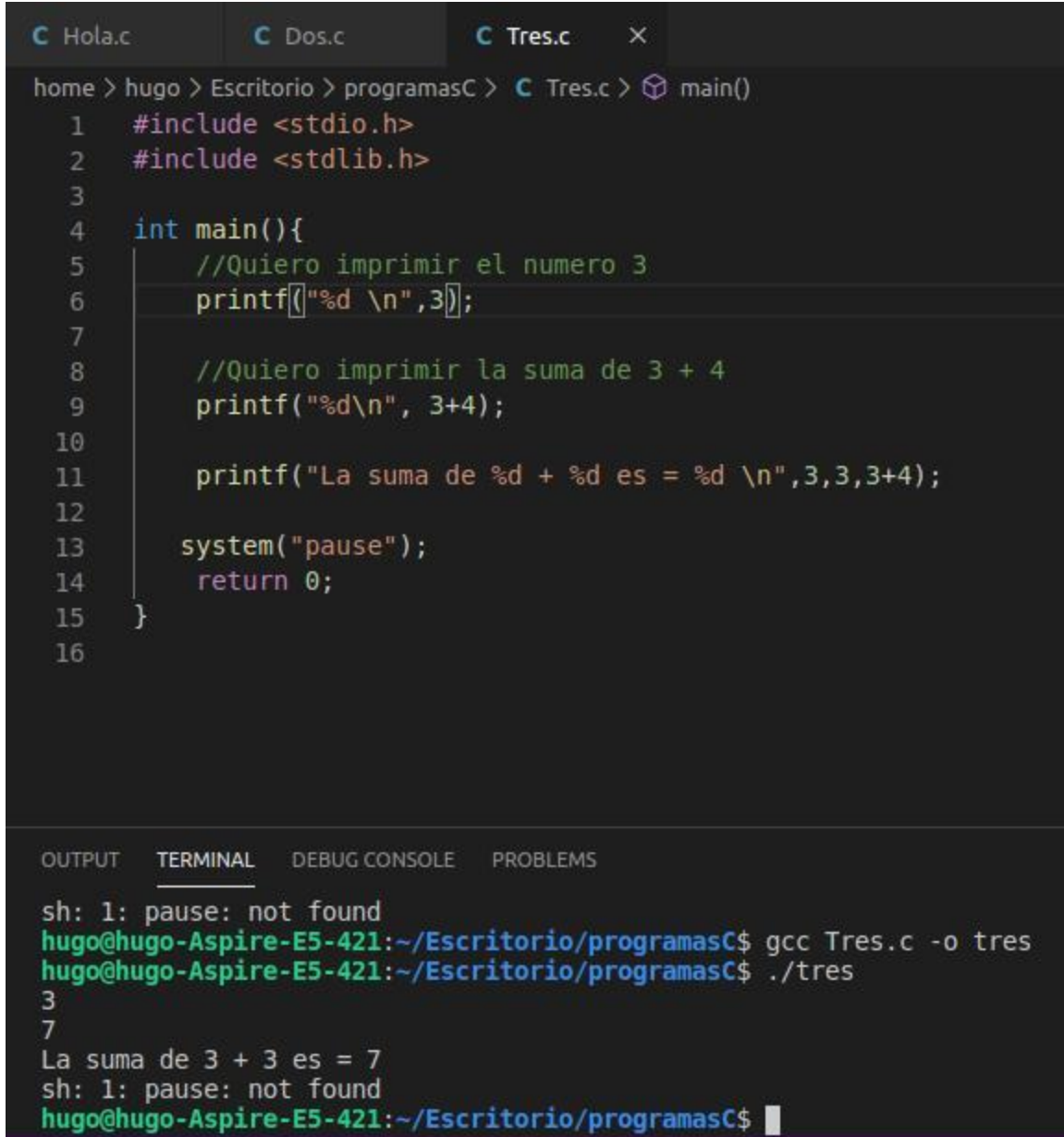
```
C Hola.c  C Dos.c  X
home > hugo > Escritorio > programasC > C Dos.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      printf("Este es un texto \n") ;
6      printf("que fue impreso desde C");
7      printf("\n");
8      printf("Ya soy todo un programador");
9      printf("\n");
10     system("pause");
11     return 0;
12 }
13

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
compilation terminated.
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Dos.c -o dos
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./dos
Este es un texto
que fue impreso desde C
Ya soy todo un programador
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```



Los estándares se ingresan, la clase principal, el printf nos sirve para incluir variables de valores enteros porque “%d” es para enteros, donde se realiza una suma de dos números, en un mensaje se realiza la suma, las variables se guardan en %d así que cada valor debe tener su respectivo espacio en memoria.

Lo compilamos y ejecutamos mostrando nuestro código, así como la suma de la operación realizada.



```
home > hugo > Escritorio > programasC > Tres.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      //Quiero imprimir el numero 3
6      printf("%d \n",3);
7
8      //Quiero imprimir la suma de 3 + 4
9      printf("%d\n", 3+4);
10
11     printf("La suma de %d + %d es = %d \n",3,3,3+4);
12
13     system("pause");
14     return 0;
15 }
16
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Tres.c -o tres
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./tres
3
7
La suma de 3 + 3 es = 7
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

En el siguiente código de igual manera, se realiza una suma tomando en cuenta lo anterior, los estándares, secuencia principal, espacio en memoria de los números



para respectivamente hacer la operación. La compilación, ejecución del programa mostrando la suma de los números.

VARIABLES C

Los estándares se incluyen, se define una variable con valor estático, se imprime un mensaje con el valor ya declarado, terminando la secuencia con una pause, la salida del mensaje.

Compilación, ejecución del programa donde nos damos cuenta que se muestra el mensaje que nos da el valor



```
C Hola.c  C Dos.c  C Tres.c  C Cuatro.c  X
home > hugo > Escritorio > programasC > C Cuatro.c > ...
1  #include <stdio.h>
2  #define PI 3.1416
3
4  int main(){
5      printf("PI vale %f",PI);
6
7
8      return 0;
9  }
10

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./tres
3
7
La suma de 3 + 3 es = 7
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Cuatro.c -o cuatro
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./cuatro
PI vale 3.141600hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

Los estándares se declaran, ahora declaramos variables con su respectivo valor, en el mensaje se realiza la suma guardando el resultado en espacio en memoria, un salto de página así mostrara la parte de la pause debajo del resultado.

En la ejecución muestra el valor de la suma.



```
C Hola.c C Dos.c C Tres.c C Cuatro.c C Cinco.c X
home > hugo > Escritorio > programasC > C Cinco.c > ...
1  #include <stdio.h>
2
3
4  int main(){
5      int a =3;
6      int b =4;
7
8      //Quiero imprimir la suma de 3 + 4
9      printf("%d", (a+b));
10
11     printf("\n");
12
13
14     return 0;
15 }
16

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
La suma de 3 + 3 es = 7
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Cuatro.c -o cuatro
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./cuatro
PI vale 3.141600hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Cinco.c -o cinco
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./cinco
7
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

Los estándares se agregan, clase principal, se declara un valor de tipo entero, la variable se ocupa de nuevo asignado un número 3, la variable vale 3 se le suma 5, así variable vale 8 y como se incrementa eso quiere decir que se le suma. Se imprime el mensaje con el resultado de la variable.

En la ejecución se imprime el resultado que es 9.



```
C Seis.c x
home > hugo > Escritorio > programasC > C Seis.c > ...
1  #include <stdio.h>
2
3
4  int main(){
5      int a;
6      a=3;
7      a=a+5;
8      a++;
9      printf("%d\n",a);
10     return 0;
11 }
12

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./cuatro
PI vale 3.141600hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Ci
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./cinco
7
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Seis.c -o seis
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./seis
9
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

El punto de este código, es obtener el área de un círculo, determinando así el valor del radio como el valor de PI, las variables se declaran como double por los puntos decimales, existe una variable con la multiplicación para obtener el valor.

Se imprime con un espacio en memoria de la variable de resultado, en la ejecución se muestra el resultado de la operación.



Los estándares, con la clase principal, donde se declaran variables de tipo double con su valor ya definido, lo que se realiza es una suma de estas dos variables, donde en la impresión se reserva un espacio en memoria para el resultado.

```
Seis.c  Siete.c  X
home > hugo > Escritorio > programasC > C Siete.c > main()
1  #include <stdio.h>
2
3
4  int main(){
5      double r= 5;
6      double pi = 3.1416;
7      double area =pi * r * r;
8
9      printf("%f \n",area);
10     return 0;
11 }
12
```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
7
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Seis.c -o seis
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./seis
9
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Siete.c -o siete
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./siete
78.540000
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

En la compilación nos dirá si existe un error de ser así no se podrá ejecutar y nos dirá donde está el error por lo que debemos de corregirlo para que este sea ejecutado. Nos damos cuenta que en la ejecución nos muestra la suma de estas dos variables.

El ejercicio nos pide que calculemos en segundos los años vividos, por lo que declaramos una variable de tipo entero donde su valor es la multiplicación de los años, por los días del año, horas, minutos segundos.



```
C Seis.c  C Siete.c  C Ocho.c  X
home > hugo > Escritorio > programasC > C Ocho.c > ...
1  #include <stdio.h>
2
3
4  int main(){
5      double a = 3.1;
6      double A = 4.5;
7      printf("%f \n",a+A);
8      return 0 ;
9  }
10

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
9
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Siete.c -o siete
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./siete
78.540000
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Ocho.c -o ocho
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./ocho
7.600000
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

En el mensaje ingresamos un texto dentro de las comillas, pero después recordemos que debe de llevar el espacio en memoria, así como el nombre de la variable. En la ejecución



nos aparece nuestro mensaje como el resultado.

```
home > hugo > Escritorio > programasC > C Edad.c > ...  
1  #include <stdio.h>  
2  
3  
4  int main(){  
5      int edad=20;  
6      edad= edad*365*24*60*60;  
7      printf("Mi edad en segundos es: %d\n",edad);  
8      return 0 ;  
9  }  
10
```



```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS  2  
hugo@hugo-Aspire-E5-421:~$ cd Escritorio/  
hugo@hugo-Aspire-E5-421:~/Escritorio$ cd programasC/  
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Edad.c -o edad  
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./edad  
Mi edad en segundos es: 630720000  
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

OPERADOR INCREMENTO

Este ejercicio nos dice que declaramos dos variables con su valor, pero en la impresión de mensaje tenemos un incremento-decremento, esto quiere decir que a la variable “var++” se suma 1, después no se le hace nada, ahora se le resta 2.

Esto nos queda de la siguiente manera en impresión, los resultados e muestra uno después de otro esto es porque no le pusimos el salto de línea dentro de las comillas que están en los mensajes.



```
C Operadores4.c  C Operadores3.c  C Operadores2.c  C Operadores1.c  C Operadores0.c X
home > hugo > Escritorio > programasC > C Operadores0.c > main()
1  #include <stdlib.h>
2  #include <stdio.h>
3  int main (){
4      int cal ;
5      puts("ingresa calificacion");
6      scanf ("%d",&cal);
7      if (cal>=8){
8          puts("Felicidades aprobaste");
9      }
10     else{
11
12         puts("reprobado");
13     }
14
15     system ("Pause");
16     return 0;
17 }
```

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
1 0
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Operadores0.c -o ope0
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./ope0
ingresa calificacion
7
reprobado
sh: 1: Pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

Estándares de entra y salida, la clase principal, se declaran variables con valor de 1, ahora se hace un post decremento que esto es antes de la variable donde su valor cambia como vale uno se le suma 1 o sea 2, pero con el otro caso se le resta entonces ahora el resultado es 0 en los dos casos.

Lo podemos ver en la ejecución del programa de igual manera los resultados están de manera consecutiva debido a no la implementación del salto de línea.



```
Operadores4.c  Operadores3.c  Operadores2.c  Operadores1.c X
home > hugo > Escritorio > programasC > C Operadores1.c > ...
1  #include <stdio.h>
2  int main(){
3      int a = 1 ;
4      int b = 1 ;
5
6      printf("%d ",a++);
7      printf("%d\n",a);
8
9      printf("%d ",b--);
10     printf("%d\n",b);
11
12
13     return 0;
14 }
15

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./ope2
2 2
0 0
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Operadores1.c -o ope1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./ope1
1 2
1 0
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

Estándares de entra y salida, la clase principal, se declara una variable de 5, ahora se hace incremento que esto es después de la variable donde su valor cambia como 5 se le suma 1 o sea 6, la otra variable vale 10, su valor en el decremento el valor es de 9.

Lo podemos ver en la ejecución del programa de igual manera los resultados están de manera consecutiva debido a no la implementación del salto de línea



```
Operadores4.c  Operadores3.c  Operadores2.c X
home > hugo > Escritorio > programasC > Operadores2.c > main()
1  #include <stdio.h>
2  int main(){
3      int a = 1 ;
4      int b = 1 ;
5
6      printf("%d ", ++a);
7      printf("%d\n", a);
8
9      printf("%d ", --b);
10     printf("%d\n", b);
11
12
13     return 0;
14 }
15

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
a : 6
d 10
--d 9 d 9
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Operadores2.c -o ope2
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./ope2
2 2
0 0
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

En este ejercicio es encontrar el valor de z por medio de variables ya declaradas ocupando el pos-tincremento y decremento por una multiplicación. Donde y vale 10 y x vale 6, mostrando los mensajes con las variables y el espacio en memoria, por el salto de línea por eso es que los números están uno debajo del otro.

En la ejecución en pantalla nos damos cuenta del resultado.



```
C Precedencia2.c  C Precedencia1.c  C Precedencia.c  C Operadores4.c X
home > hugo > Escritorio > programasC > C Operadores4.c > main()
1  #include <stdio.h>
2  int main(){
3
4      int a = 5 ;
5      int d = 10;
6      int z = ++a * d-- ;
7
8      printf("a : %d\n",a);
9      printf("d : %d\n",d);
10     printf("z : %d\n",z);
11
12     return 0;
13 }
14
15

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
resultado 9.285714
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Operadores4.c -o ope4
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./ope4
a : 6
d : 9
z : 60
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

PRECEDENCIA DE OPERADORES

En este ejercicio declaramos variables de tipo entero con su valor, a z se le asigna operaciones de acuerdo a las variables donde hay multiplicaciones, resta, división. Para esto tomemos en cuenta la prioridad de los signos para obtener el resultado



```
Precedencia2.c  Precedencia1.c  Precedencia.c X
home > hugo > Escritorio > programasC > C Precedencia.c > main()
14  printf("El resultado es : %4f\n", resul);
15
16  double un = 5;
17  double dus = 2;
18  double tre = 1;
19  double cua = 4;
20  double ope1;
21  double ope2;
22  double ope3;
23  double ope4;
24
25  ope1= (dus)-(tre/cua);
26  ope2= (tre)+(dus/ope1);
27  ope3= (tre/ope2);
28  ope4= (un)+(dus/ope3);
29
30  printf("resultado  %2f\n", ope4);
31
32  system("pause");
33  return 0;
34

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./prel
z : -6
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Precedencia.c -o pre
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./pre
El resultado es : 0.976812
resultado  9.285714
sh: 1: pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

En la ejecución en pantalla nos damos cuenta de ello.

En este código hablamos más de los operadores y de la función de los paréntesis, donde modifican mucho los resultados, tenemos el salto de línea, reserva el espacio en memoria, en la ejecución nos damos cuenta.

El ejercicio consta de tener muy en cuenta los paréntesis, declaramos los valores de las variables para que los valores los tome de la variable, es suma, división de fracciones.

```
C Precedencia2.c C Precedencia1.c X
home > hugo > Escritorio > programasC > C Precedencia1.c > main()
1  #include <stdio.h>
2  int main(){
3
4      int p = 5;
5      int q = 1;
6      int r = 2;
7      int w = 3;
8      int x = 9;
9      int y = 6;
10     int z = p * r % q + w / x - y ;
11     printf("z : %d\n",z);
12
13     return 0;
14 }
15
16

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./pre1
1 + 2 : 7
1 + 2 : 9
1 + 2 : 7
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Precedencia1.c -o pre1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./pre1
z : -6
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

En el desarrollo del código nos damos cuenta de como funcionan los paréntesis ya que primero se realiza lo que está dentro de ellos. En la ejecución nos da el valor de las operaciones, en decimal por el tipo de dato que es un double.



```
C Precedencia2.c X
home > hugo > Escritorio > programasC > C Precedencia2.c > ...
1  #include <stdio.h>
2  int main(){
3      printf(" 1 + 2 : %d\n",1+2*3);
4      printf(" 1 + 2 : %d\n",(1+2)*3);
5      printf(" 1 + 2 : %d\n",1+(2*3));
6
7      return 0;
8  }
9
10

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS
false ^false : 0n
sh: 1: Pause: not found
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Precedencia2.c -o pre1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./pre1
 1 + 2 : 7
 1 + 2 : 9
 1 + 2 : 7
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

OPERADORES LÓGICOS Y DE RELACIÓN

El operador lógico funciona de esa manera donde cada uno nos dará un resultado diferente de acuerdo a si es un or, y, xor, negación.

En la ejecución nos podemos dar cuenta como es que funciona en el y, solo uno puede ser verdadero, en el or, al menos un verdadero nos da verdadero, en el xor solo debe de existir un verdadero.



```
C Logicos4.c C Logicos3.c C Logicos2.c C Logicos1.c X
home > hugo > Escritorio > programasC > C Logicos1.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5      printf("***and***\n");
6      printf("true && true : %d \n", (1&&1));
7      printf("true && false : %d \n", (1&&0));
8      printf("false && true : %d \n", (0&&1));
9      printf("false && false : %d \n", (0&&0));
10
11     printf("***or***\n");
12     printf("true || true : %d \n", (1||1));
13     printf("true || false : %d \n", (1||0));
14     printf("false || true : %d \n", (0||1));
15     printf("false || false : %d \n", (0||0));
16
17     printf("***xor***\n");
18     printf("true ^ true : %d \n", (1^1));
19     printf("true ^ false : %d \n", (1^0));
20     printf("false ^ true : %d \n", (0^1));
21     printf("false ^ false : %d \n", (0^0));
22 }
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Logicos1.c -o log1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./log1
***and***
true && true : 1
true && false : 0
false && true : 0
false && false : 0
***or***
true || true : 1
true || false : 1
false || true : 1
false || false : 0
***xor***
true ^ true : 0
true ^ false : 1
false ^ true : 1
false ^ false : 0
```

Ocupamos los estándares de entrada y salida, así como la clase principal, el espacio en memoria y el salto de línea.

Tenemos otro ejercicio de igual manera donde el true=1 y false=0, donde ocupamos operadores lógicos de manera diferente.



```
C Logicos4.c  C Logicos3.c  C Logicos2.c X
home > hugo > Escritorio > programasC > C Logicos2.c > ...
1  # include <stdio.h>
2  # include <stdlib.h>
3  int main(){
4      int p=1;
5      int q=0;
6      int r=1;
7      int t=0;
8      printf("%d \n", (p&& r));
9      printf("%d \n", (q||t));
10     printf("%d \n", (p&& q||r&& t));
11     printf("%d \n", (p^q^r^t));
12     printf("%d \n", (!q&&!t));
13     printf("%d \n", (!!!p));
14
15     return 0;
16 }
17

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Logicos2.c -o log2
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./log2
1
0
0
0
1
0
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

Definimos las variables su valor como el tipo de dato, de nuevo ocupamos los operadores lógicos y usamos paréntesis.

En la ejecución nos damos cuenta como el valor cambia de acuerdo a lo que se pide, bueno de acuerdo a los operadores lógicos



```
C Logicos4.c  C Logicos3.c X
home > hugo > Escritorio > programasC > C Logicos3.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(){
5
6      printf("%d \n", (3>5));
7      printf("%d \n", (3<5));
8      printf("%d \n", (3==5));
9      printf("%d \n", (3!=5));
10
11     return 0;
12 }
13
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS

```
1
1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Logicos3.c -o log3
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./log3
0
1
0
1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```



```
Logicos4.c X
home > hugo > Escritorio > programasC > C Logicos4.c > ...
1  # include <stdio.h>
2  # include <stdlib.h>
3  int main(){
4      int w=9;
5      int x=3;
6      int y=7;
7      int z=-2;
8
9
10     printf("%d \n", (x<y&&w>x));
11     printf("%d \n", (x>=w^z==y));
12     printf("%d \n", (y<=x||x!=w));
13     printf("%d \n", (w==9^x==3));
14     printf("%d \n", (y>z&&z<x));
15     printf("%d \n", (!w!=9));
16
17
18     return 0;

OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS

hugo@hugo-Aspire-E5-421:~/Escritorio$ cd programasC/
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ gcc Logicos4.c -o log4
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$ ./log4
1
0
1
0
1
1
hugo@hugo-Aspire-E5-421:~/Escritorio/programasC$
```

