

Operadores y expresiones

🕒 Created	@May 11, 2023 9:24 PM
🏷 Tags	

Operadores de incrementación y decrementación

Los operadores ++ y — suman o restan 1 a su argumento, respectivamente, cada vez que se aplique a una variable.

Entonces:

a++

es igual que,

a = a+1

Tienen la propiedad de usarse como prefijo o sufijo, el resultado de la expresión puede variar depende del contexto.

++n;

n++;

Formas de escribirse

Incrementación	Decrementación
++n	—n
n += 1	n -= 1
n = n + 1	n = n - q

Son iguales como:

—n;

n—;

Cambia cuando están en expresiones como;

```
m= n++;  
printf("n = %d", n--);
```

Es distinto a si se usa como prefijo;

```
m = ++n;
printf("n = %d", --n);
```

The screenshot shows a C++ program in Sublime Text with the following code:

```
1 #include<stdio.h>
2 //la misma variable se puede manipular varias veces dentro de un mismo programa
3 int main(){
4     int n=9;
5     int m;
6
7     m = n++;
8     printf("n = %d", m);
9
10    m = ++n;
11    printf("n = %d", m);
12
13    return 0;
14 }
15
```

The terminal output shows the execution of the program:

```
yareli@yareli: ~/Documentos/C/libro/opyex on 0 master [?] took 6ms
$ g++ operadores.cpp -o op
yareli@yareli: ~/Documentos/C/libro/opyex on 0 master [?] took 148ms
$ ./op
n = 9n = 11
yareli@yareli: ~/Documentos/C/libro/opyex on 0 master [?] took 5ms
$
```

Así se ve imprimiendo solo la m.

The screenshot shows a C++ program in Sublime Text with the following code:

```
1 #include<stdio.h>
2 //la misma variable se puede manipular varias veces dentro de un mismo programa
3 int main(){
4     int n=9;
5     int m;
6
7     m = n++;
8     printf("n = %d", n--);
9
10    m = ++n;
11    printf("n = %d", --n);
12
13    return 0;
14 }
15
```

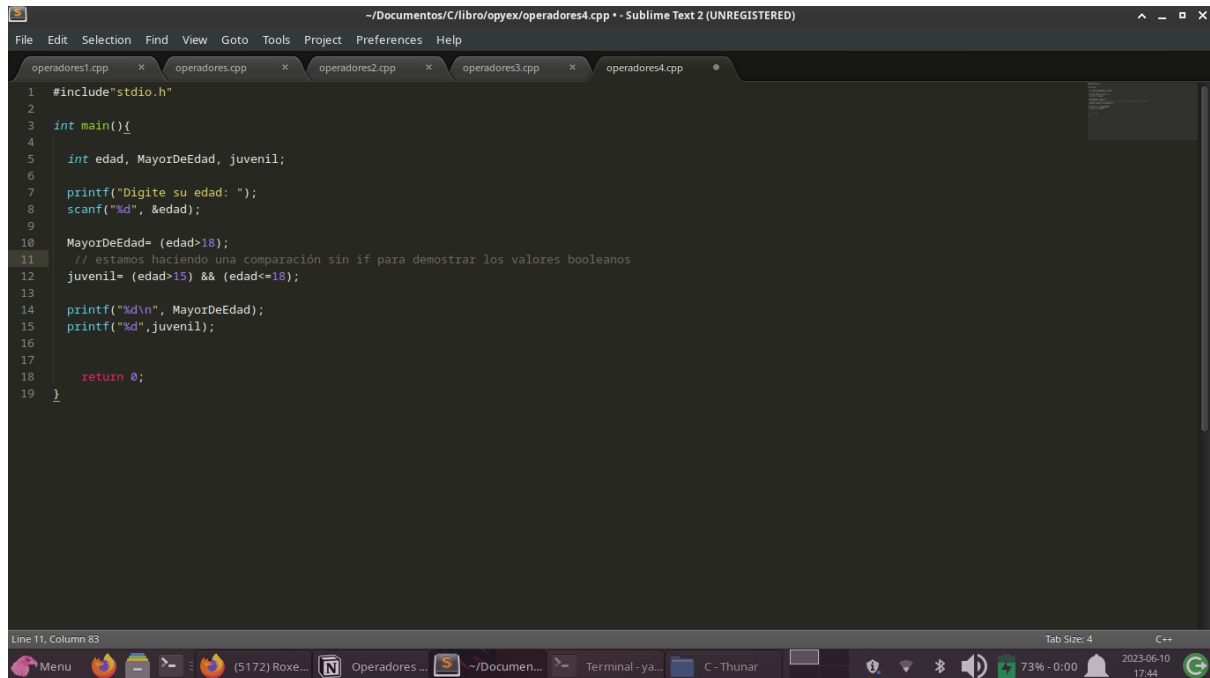
The terminal output shows the execution of the program:

```
yareli@yareli: ~/Documentos/C/libro/opyex on 0 master [?] took 5ms
$ g++ operadores.cpp -o op
yareli@yareli: ~/Documentos/C/libro/opyex on 0 master [?] took 168ms
$ ./op
n = 10n = 9
yareli@yareli: ~/Documentos/C/libro/opyex on 0 master [?] took 11ms
$
```

Así se ve imprimiendo la n decrementada tanto en prefijo como en sufijo.

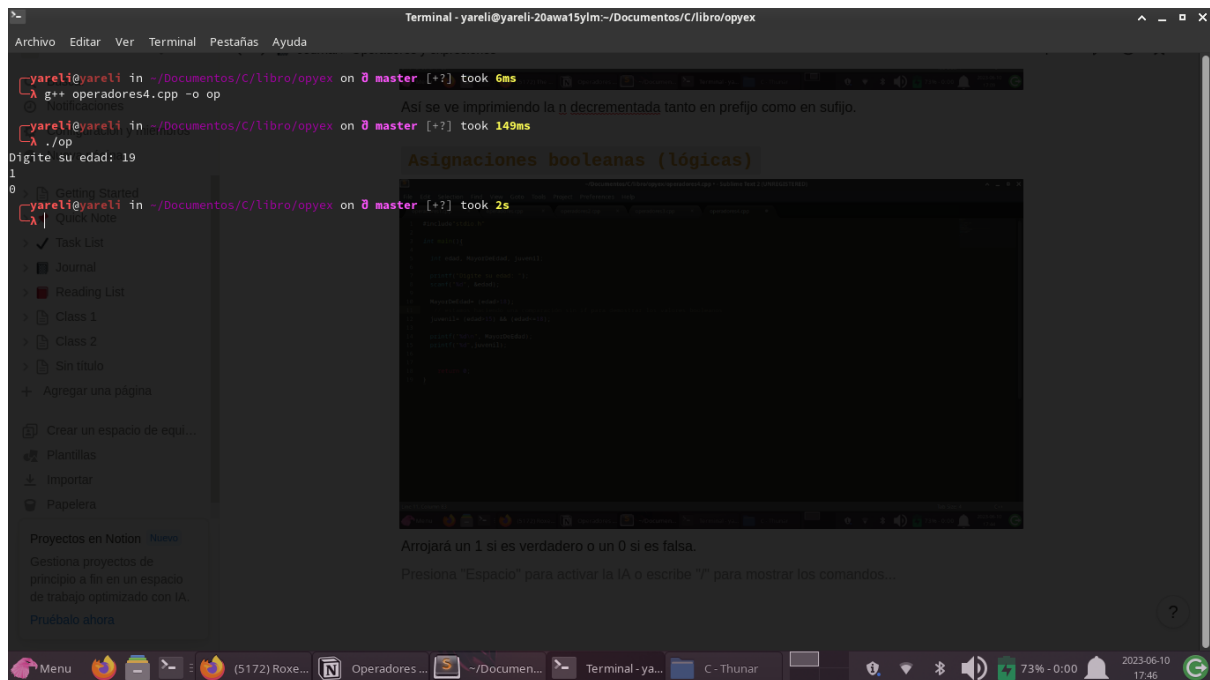
Asignaciones booleanas (lógicas)

Las comparaciones lógicas no siempre son necesarias con "if", en este caso para hacer asignaciones booleanas.



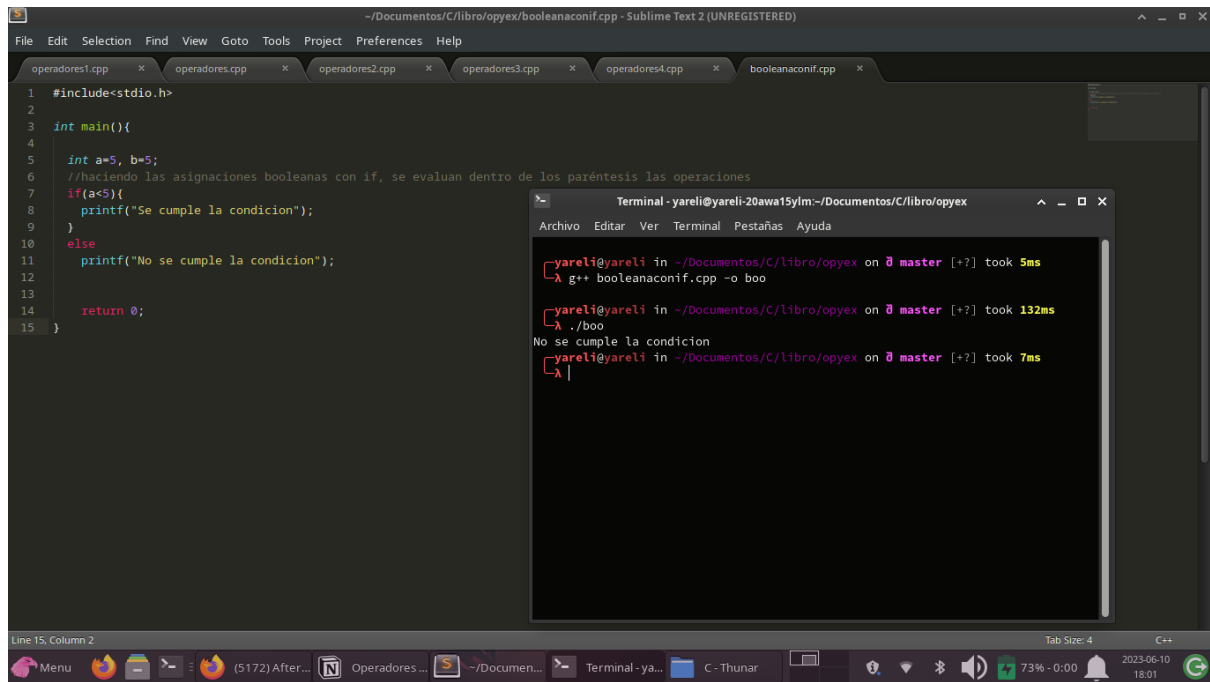
```
1 #include "stdio.h"
2
3 int main(){
4
5     int edad, MayorDeEdad, juvenil;
6
7     printf("Digite su edad: ");
8     scanf("%d", &edad);
9
10    MayorDeEdad= (edad>18);
11    // estamos haciendo una comparación sin if para demostrar los valores booleanos
12    juvenil= (edad>15) && (edad<=18);
13
14    printf("%d\n", MayorDeEdad);
15    printf("%d", juvenil);
16
17    return 0;
18 }
19 }
```

Arrojará un 1 si es verdadero o un 0 si es falsa.



```
yareli@yareli: ~/Documentos/C/libro/opyex
$ g++ operadores4.cpp -o op
Así se ve imprimiendo la n decrementada tanto en prefijo como en sufixo.
yareli@yareli: ~/Documentos/C/libro/opyex
$ ./op
Digite su edad: 19
1
1
```

En la primera se pedía ver si era mayor de edad, **como ingresé "19", se arrojó el 1 dando a entender que es verdadero.**



```
#include<stdio.h>

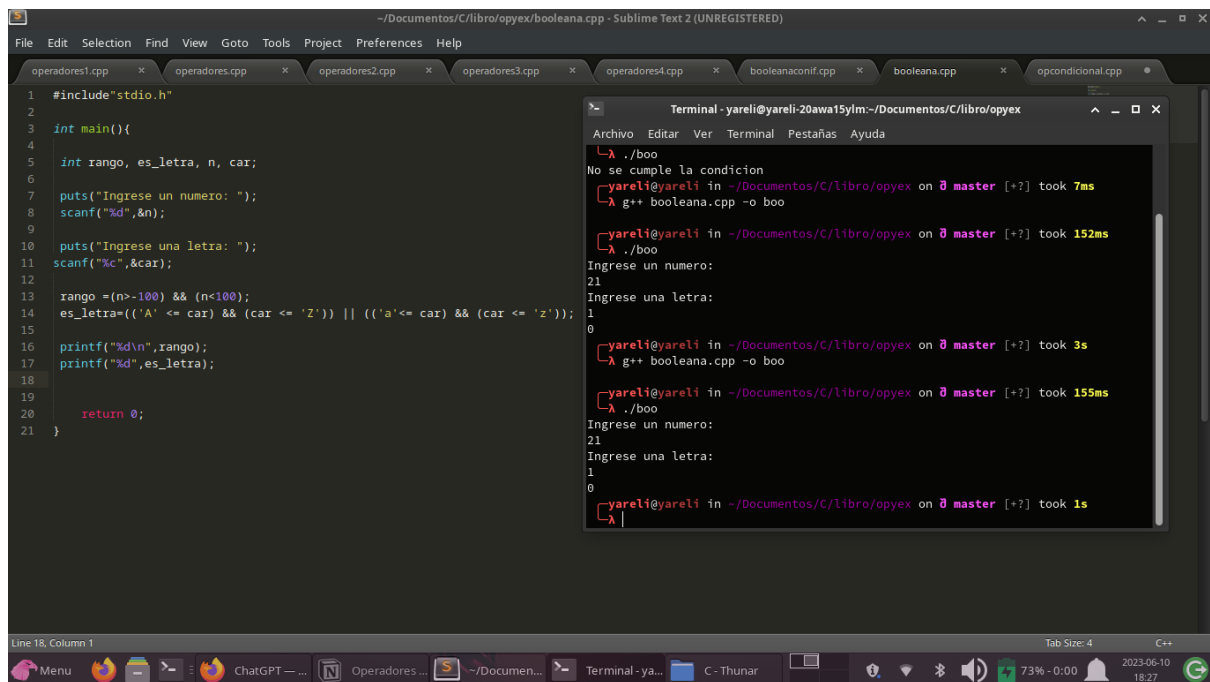
int main(){
    int a=5, b=5;
    //haciendo las asignaciones booleanas con if, se evaluan dentro de los paréntesis las operaciones
    if(a<5){
        printf("Se cumple la condicion");
    }
    else
        printf("No se cumple la condicion");

    return 0;
}
```

```
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 5ms
λ g++ booleanaonif.cpp -o boo
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 132ms
λ ./boo
No se cumple la condicion
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 7ms
λ |
```

Aquí ya empleamos un if normal, para prácticamente hacer lo mismo.

Este es otro ejemplo, sin embargo, falta averiguar cómo tomar en cuenta el segundo valor que quiero que se compare en la segunda comparación.



```
#include<stdio.h>

int main(){
    int rango, es_letra, n, car;

    puts("Ingrese un numero: ");
    scanf("%d",&n);

    puts("Ingrese una letra: ");
    scanf("%c",&car);

    rango =(n>=100) && (n<100);
    es_letra=(( 'A' <= car) && (car <= 'Z')) || (( 'a' <= car) && (car <= 'z'));

    printf("%d\n",rango);
    printf("%d",es_letra);

    return 0;
}
```

```
λ ./boo
No se cumple la condicion
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 7ms
λ g++ booleana.cpp -o boo
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 152ms
λ ./boo
Ingrese un numero:
21
Ingrese una letra:
1
0
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 3s
λ g++ booleana.cpp -o boo
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 155ms
λ ./boo
Ingrese un numero:
21
Ingrese una letra:
1
0
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 1s
λ |
```

Operador condicional ?:

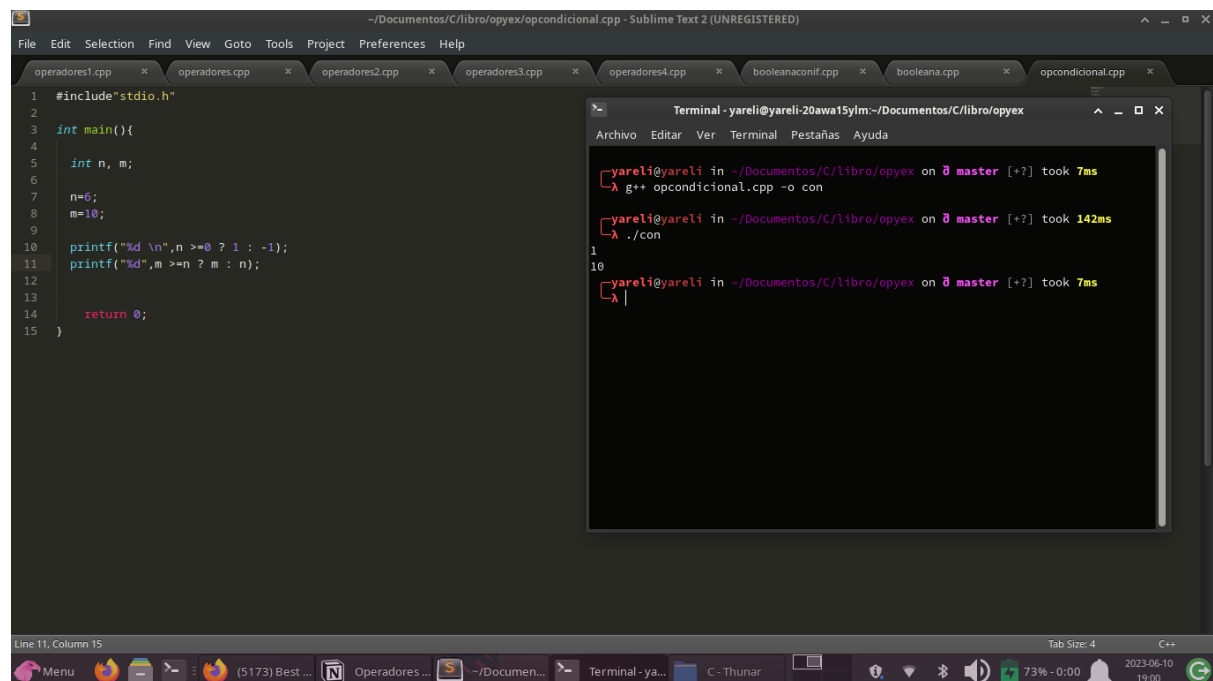
Este operador se usa para reemplazar a la sentencia **if-else** lógica en algunas expresiones. El formato es:

```
expresion_c ? expresion_v : expresion_f;
```

Ahora, ¿cómo se lee esto?, bueno; se dice que **expresion_c** es la “sentencia” como sería en el if, entonces si es verdadero, arrojará el valor de la **expresion_v**, en caso contrario, arrojará el valor de la expresión **expresion_f**.

1er Ejemplo sencillo.

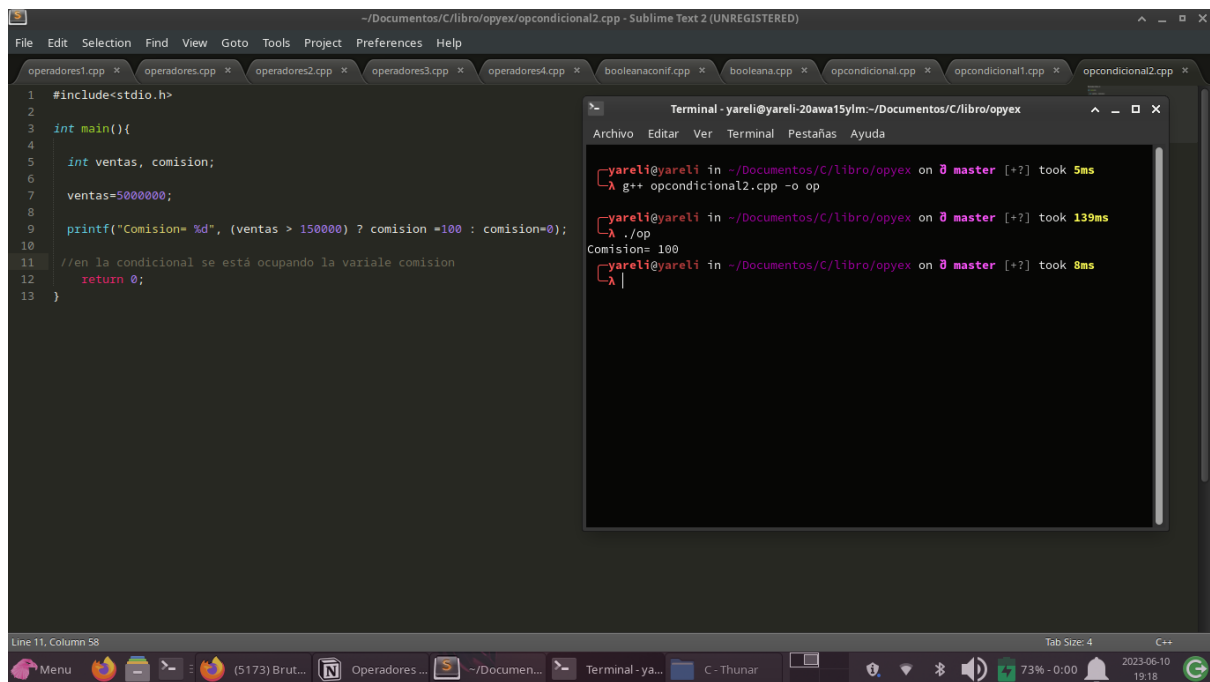
Nótese que dentro del printf ponemos que se imprimirá un número, dicho número será por la condición que ponemos y la asignamos con una coma después de las comillas del printf.



```
#include "stdio.h"
int main(){
    int n, m;
    n=6;
    m=10;
    printf("%d\n", n >= 0 ? 1 : -1);
    printf("%d", m >= n ? m : n);
    return 0;
}
```

```
yareli@yareli: ~/Documentos/C/libro/opyex on 8 master [?] took 7ms
λ g++ opcondicional.cpp -o con
yareli@yareli: ~/Documentos/C/libro/opyex on 8 master [?] took 142ms
λ ./con
1
10
yareli@yareli: ~/Documentos/C/libro/opyex on 8 master [?] took 7ms
λ
```

Ejemplo, usando dos variables, es decir que dentro de la condición se le asignó el valor a la segunda variable, no como en el primer ejemplo que desde el principio se les asignó valor a ambas variables y los datos a arrojar eran número que dábamos.



```
#include<stdio.h>

int main(){
    int ventas, comision;
    ventas=5000000;
    printf("Comision= %d", (ventas > 150000) ? comision =100 : comision=0);
    //en la condicional se está ocupando la variable comision
    return 0;
}
```

```
Terminal - yareli@yareli-20awa15ylm:~/Documentos/C/libro/opyex
Archivo Editar Ver Terminal Pestañas Ayuda

yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 5ms
λ g++ opcondicional2.cpp -o op

yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 139ms
λ ./op
Comision= 100

yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [+?] took 8ms
λ |
```

Operadores especiales: (), []

El operador ()

Este operador se ocupa en funciones, sirve para encerrar los argumentos de una función, efectuar conversaciones explícitas de tipo, indicar en el seno de una declaración que un identificador corresponde a una función y resolver los conflictos de prioridad entre operadores.

El operador []

Sirve para **dimensionar los arreglos** (arrays) y designar un elemento de un arreglo (array).

```
#include<stdio.h>

int main(){
    double v[20];
    printf("v[2] = %e",v[2]);
    return 0;
}
```

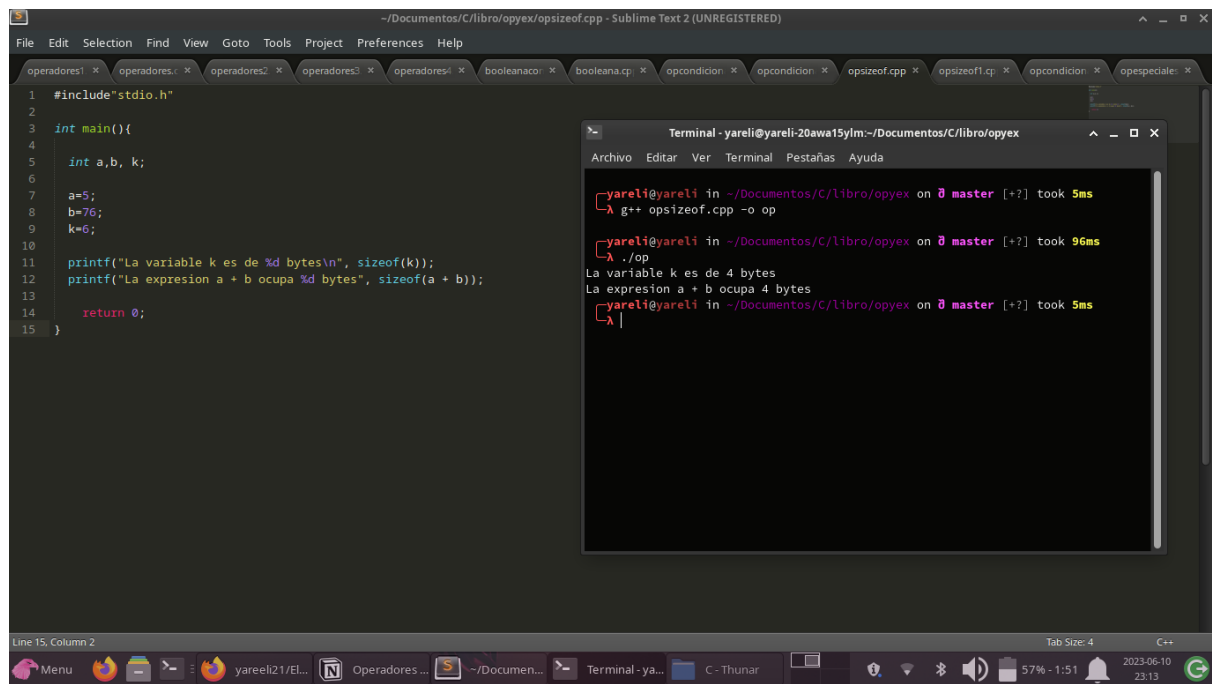
```
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 4ms
λ g++ opespeciales.cpp -o op
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 100ms
λ ./op
v[2] = 6.920093e-310
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 5ms
λ |
```

Operador sizeof

```
#include<stdio.h>

int main(){
    printf("El tamaño de variables de coma flotante es %d\n", sizeof(float));
    printf("El tamaño de variables de doble precision es %d\n", sizeof(double));
    return 0;
}
```

```
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 5ms
λ g++ opsizeof1.cpp -o op
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 98ms
λ ./op
El tamaño de variables de coma flotante es 4
El tamaño de variables de doble precision es 8
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 5ms
λ |
```



The image shows a Sublime Text 2 editor window with a C++ file named `opsizEOF.cpp`. The code defines a `main` function that declares variables `a`, `b`, and `k`, assigns values to `a` and `b`, and prints the size of `k` and the expression `a + b` using `sizeof`. A terminal window is open, showing the compilation of the program with `g++ opsizEOF.cpp -o op` and its execution with `./op`. The output of the program is displayed in the terminal.

```
#include<stdio.h>

int main(){
    int a,b, k;

    a=5;
    b=76;
    k=6;

    printf("La variable k es de %d bytes\n", sizeof(k));
    printf("La expresion a + b ocupa %d bytes", sizeof(a + b));

    return 0;
}
```

```
yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 5ms
g++ opsizEOF.cpp -o op

yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 96ms
./op
La variable k es de 4 bytes
La expresion a + b ocupa 4 bytes

yareli@yareli in ~/Documentos/C/libro/opyex on 0 master [??] took 5ms
```