



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación
salas A y B

Profesor: Alejandro Esteban Pimentel Alarcon

Asignatura: Fundamentos de programación.

Grupo: 3

No de Práctica(s): 7

Integrante(s): Ortiz Garcia Cesar Alan

*No. de Equipo de cómputo
empleado:* 30

No. de Lista o Brigada: 9070

Semestre: 20-21

Fecha de entrega: 3/oct/19

Observaciones: En esta práctica hace falta evidencias (capturas)
de la compilación y ejecución correcta del
segundo programa (el de lectura y escritura)

CALIFICACIÓN: **9**

Fundamentos de C

Aprender los fundamentos de un lenguaje de programación es como abrir las puertas a la oportunidad y a la aventura. Y, en estos tiempos, aprender a programar, aún a un nivel básico, es bastante beneficioso, ya que el lenguaje C proporciona una base para la comprensión de los conceptos básicos de programación. Si sabe cómo programar en C, puede aprender C++, C#, Java, y muchos otros lenguajes de programación.

Tipos de Datos

El compilador de C reconoce unos tipos de datos estándar como enteros, flotante y carácter. Estos tipos de datos son:

- **char**: ocupa un byte en memoria, se suele utilizar para almacenar caracteres, pues el [ASCII-E](#) usa exactamente 8 bits para representar un carácter. (-128 a +127 ó 0 a 255).
- **int**: ocupa 4 bytes y es actualmente el tamaño de la palabra de un ordenador de 32 bits, salvo que se esté en posesión de un ordenador de 64 bits con un SO que lo soporte (-2^{31} a $2^{31}-1$ ó 0 a $2^{32}-1$).
- **long** ó **long int**: generalmente ocupa dos palabras (64 bits u 8 bytes) pero depende del ordenador.
- **float**: ocupa una palabra y se usa para representar números reales dado que su rango es mucho mayor, usaremos este tipo para la representación de números reales.
- **double**: ocupa dos palabras y tiene una capacidad mucho mayor que un float.
- **short**: ocupa dos bytes en memoria, lo usaremos exclusivamente cuando vayamos a necesitar grandes cantidades de memoria y nos sobre precisión, si no, utilizaremos o el int o char, por cuestiones de arquitectura del ordenador (-2^{15} a $2^{15}-1$ ó 0 a $2^{16}-1$).
- **unsigned**: se usa para especificar que la variable no tiene signo, por lo que "aumenta" su capacidad en cuanto a número positivos.
- **void**: esto no es un tipo de dato en sí mismo, pero se usa para determinar que una función no recibe parámetros o no devuelve un resultado. No pueden existir variables del tipo *void*.

• Tipo	Rango	Bytes
char	-128 ... 127 (ASCII)	1
int	-32.768 ... 32.767	4
long	-2.147.483.648 ... 2.147.483.647	8
float	$3.4 \cdot 10^{-38}$... $3.4 \cdot 10^{38}$	4
double	$1.7 \cdot 10^{-308}$... $1.7 \cdot 10^{308}$	8
void	Valor nulo	nulo

la sintaxis para declarar variables o constantes con los diferentes tipos de datos es:

```
tipo_de_dato nombre_variable;
```

Operadores

Los programas en C constan de datos, sentencias y expresiones. Una expresión es una ecuación matemática, tal como $25 * 13$. en esta expresión el símbolo (*) es el operador de multiplicación y los números 25 y 13 se llaman operandos.

El lenguaje C soporta diferentes tipos de operadores: *aritméticos*, *lógicos*, y *relacionales*.

Operador	Simbolo	Ejemplo	Significado
Suma	+	$a + b$	a más b
Resta	-	$a - b$	a menos b
Multiplicación	*	$a * b$	a por b
División	/	a / b	a dividido b
Residuo	%	$a \% b$	a residuo de b
Signo (negativo)	-	-a	a negativo

Operadores Relacionales

Son aquellos operadores que se utilizan para la toma de decisiones que se puedan necesitar dentro de un programa.

Operador	Simbolo	Ejemplo	Significado
Igual	==	$x == y$	x es igual a y
Diferente	!=	$x != y$	x es diferente de y
Mayor que	>	$x > y$	x es mayor que y

Mayor o igual que	>=	$x \geq y$	x es mayor o igual que y
Menor que	<	$x < y$	x es menor que y
Menor o igual que	<=	$x \leq y$	x es menor o igual que y

Operadores Lógicos

Son operadores usados para realizar conectividad lógica en las expresiones.



Nota: C interpreta cualquier número entero distinto de 0 (cero) como verdadero, ya sea positivo como negativo, por lo que para usar el tipo de dato *boolean* no definido en C podremos usar tanto un *int* como un *char* (el *short* no es aconsejable usarlo salvo problemas de memoria, cosa que a partir del año 2000 no suele ocurrir salvo al programar algunos microprocesadores).

Operador	Simbolo	Ejemplo	Significado
Y (AND)	&&	$(a > b) \ \&\& \ (c < d)$	a es mayor que b y c es menor que d
O (OR)		$(a > b) \ \ (c < d)$	a es mayor que b o c es menor que d
NEGACION (NOT)	!	$!(a > b)$	a no es mayor que b

Operadores de Incremento o Decremento

Estos operadores permiten incrementar o decrementar en una unidad el valor de una variable

Ejemplo	Significado
Variable++	El valor de la variable incrementa después de una operación
++Variable	El valor de la variable incrementa antes de una operación
Variable--	El valor de la variable disminuye después de una operación
--Variable	El valor de la variable disminuye antes de una operación

También es posible crear variables de incremento o decremento mayores a la unidad, así:

Ejemplo	Significado
Variable+=3	La variable incrementa su valor en 3 unidades
Variable-=5	La variable decrementa su valor en 5 unidades

Precedencia de Operadores

La precedencia de los operadores determina el orden en que se ejecutaran las operaciones dentro de las expresiones.

Lo siguiente lista todos los operadores del C en orden descendente de precedencia. Los operadores en el mismo renglón tienen la misma precedencia.

Nivel 1: () [] ->

Nivel 2: ! ~ ++ -- *(indireccion) &(direccion de) +(unario) -(unario)

Nivel 3: *(multiplicación) / %

Nivel 4: + -

Nivel 5: << >>

Nivel 6: < <= => >

Nivel 7: == !=

Nivel 8: &(AND a nivel d

1._Mostrar y Leer

```

Suecia44:Documents fp03alu37$ gcc trabajo.c -o main_
trabajo.c:25:2: warning: implicit declaration of function 'printf' is invalid in
C99 [-Wimplicit-function-declaration]
  printf("Tu entero: %i\n", numeroEntrada);
  ^
1 warning generated.
Undefined symbols for architecture x86_64:
  "_printf", referenced from:
    _main in trabajo-c3fa27.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
Suecia44:Documents fp03alu37$ gcc trabajo.c -o main_
trabajo.c:27:2: warning: implicit declaration of function 'printf' is invalid in
C99 [-Wimplicit-function-declaration]
  printf("Tu entero: %i\n", numeroEntrada);
  ^
1 warning generated.
Undefined symbols for architecture x86_64:
  "_printf", referenced from:
    _main in trabajo-5a4bde.o
ld: symbol(s) not found for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
Suecia44:Documents fp03alu37$ gcc trabajo.c -o main_
Suecia44:Documents fp03alu37$

```

```

1 warning generated.
Undefined symbols for architecture x86_64:
  "_printf", referenced from:

```

```

trabajo.c
UNREGISTERED

4
5 int main() {
6
7     //Declarar variables al leer
8     int numeroEntrada;
9     double realEntrada;
10
11     // Asignamos variables
12     int numeroEntero = 32768;
13     char caracter = 'B';
14     float numeroReal = 89.8;
15
16     // Mostramos texto y valores
17     printf("Primer texto solo\n");
18     printf("Luego podemos poner un entero:%i\n", numeroEntero);
19     printf("Tambien podemos poner un caracter:%c\n", caracter);
20     printf("Y un numero real :%.2f\n", numeroReal);
21
22     // Leemos valores
23     scanf("%i", &numeroEntrada);
24     scanf("%lf", &realEntrada);
25
26     // Y ahora podemos mostrarlos tambien
27     printf("Tu entero: %i\n", numeroEntrada);
28     printf("Tu real: %.3lf\n", realEntrada);
29
30     return 0;
31 }

```

Line 27, Column 10 Tab Size: 4 C

is invalid in

Operadores lógicos

UNREGISTERED

```
#include <stdio.h>

int main() {

    int num1, num2, res;
    char c1, c2;

    num1 = 7;
    num2 = 15;
    c1 = 'h';
    c2 = 'H';

    printf("%i num1 es menor a num2 ? -> %i\n", num1 < num2);
    printf("%i c1 es igual a c2 ? -> %i\n", c1 == c2);
    printf("%i c1 es diferente a c2 ? -> %i\n", c1 != c2);

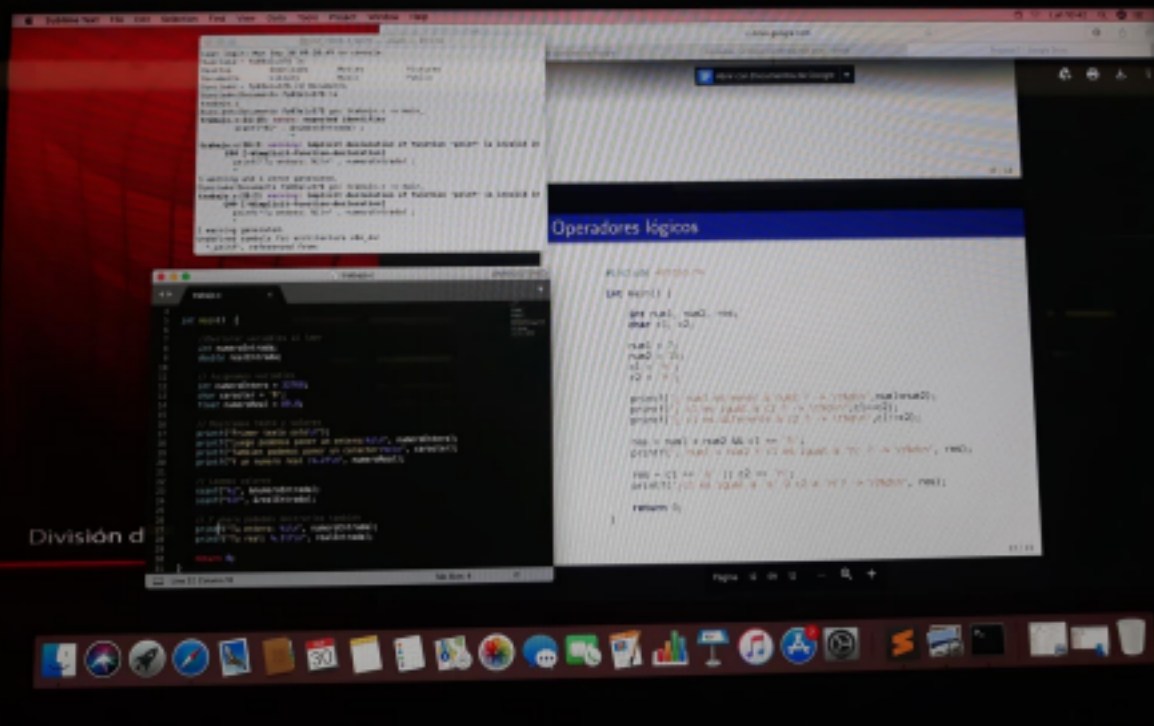
    res = num1 < num2 && c1 == 'h';
    printf("%i num1 < num2 Y c1 es igual a 'h' ? -> %i\n", res);

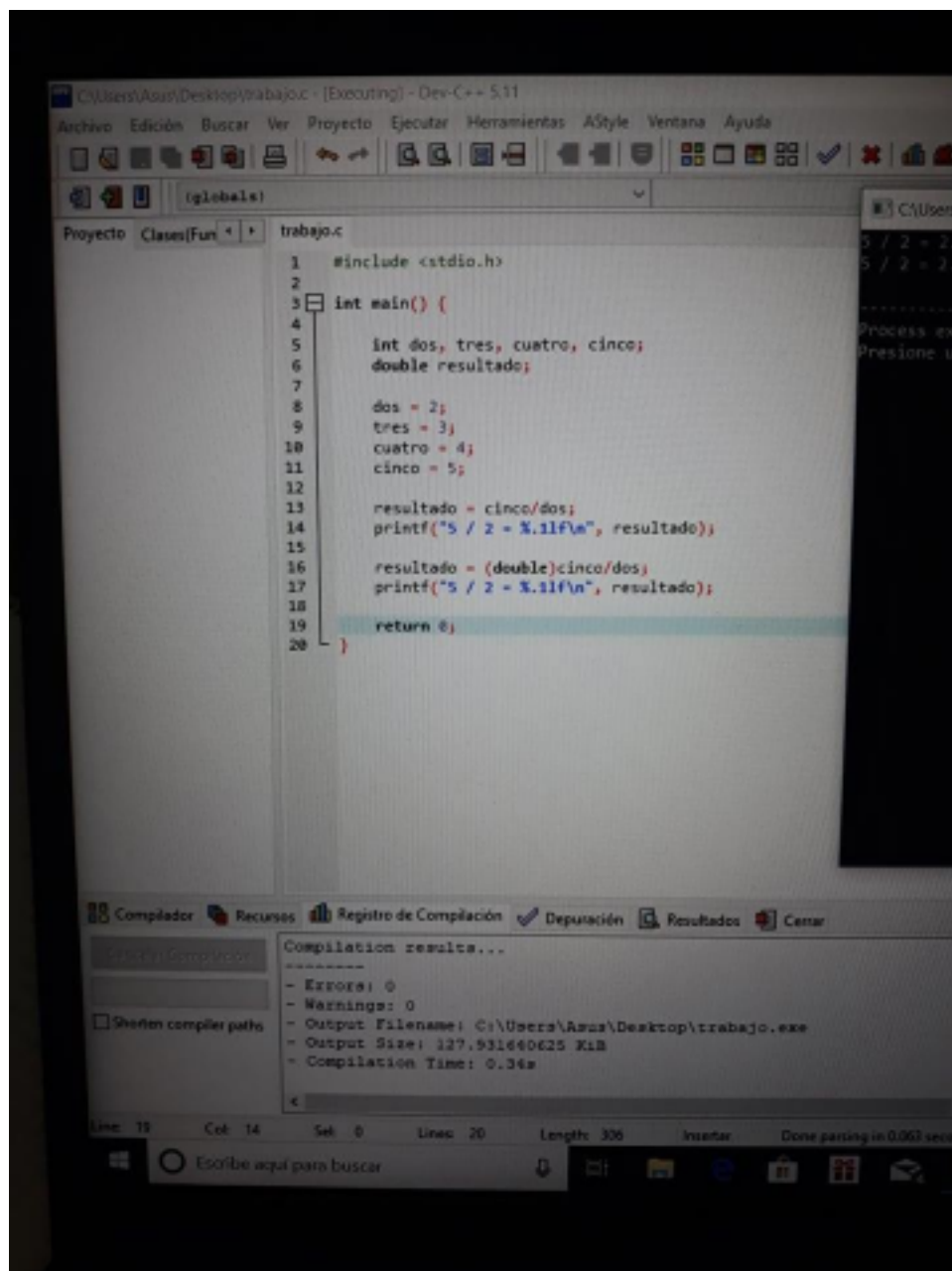
    res = c1 == 's' || c2 == 'H';
    printf("%i c1 es igual a 's' O c2 a 'H' ? -> %i\n", res);

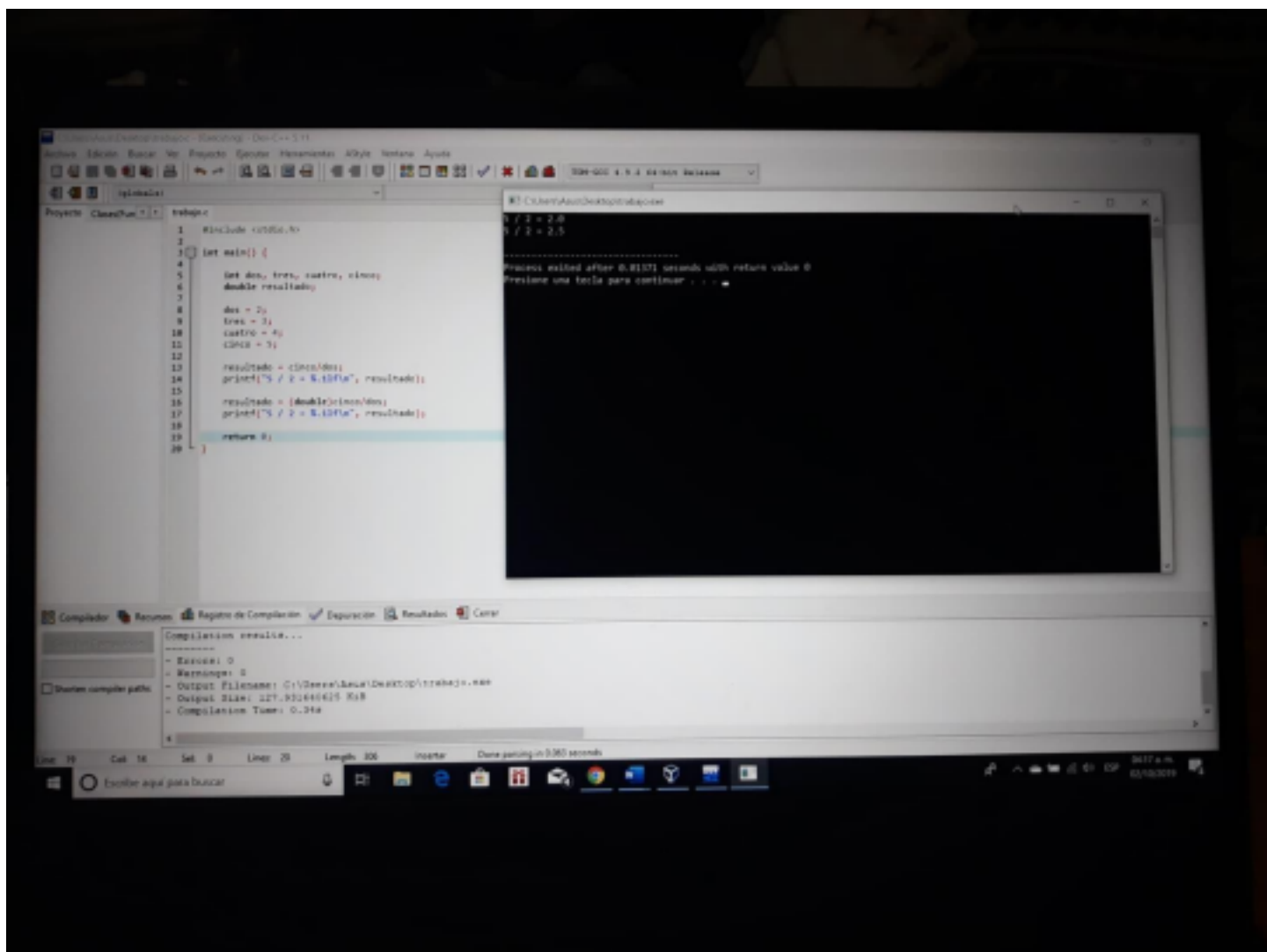
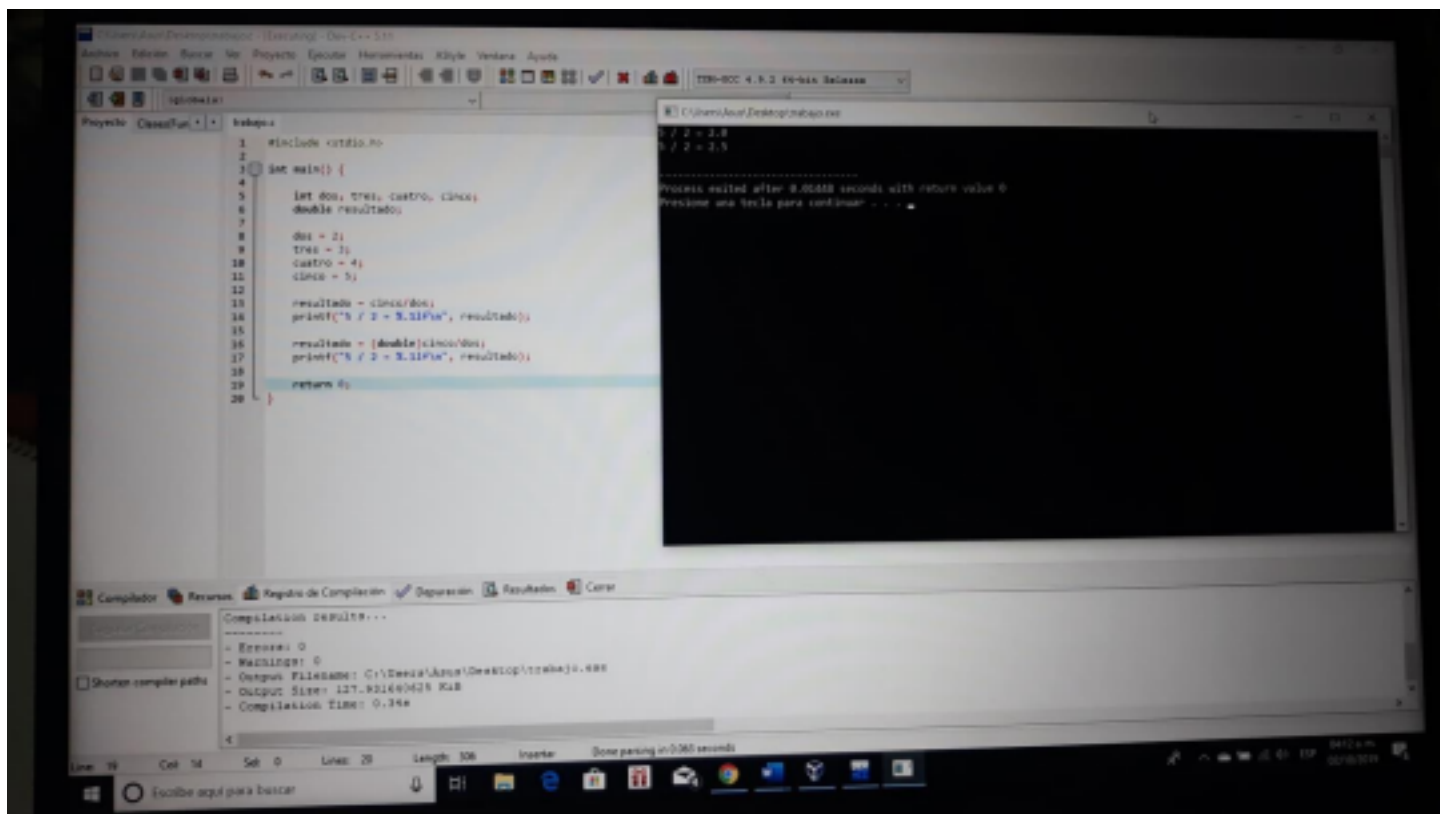
    return 0;
}
```

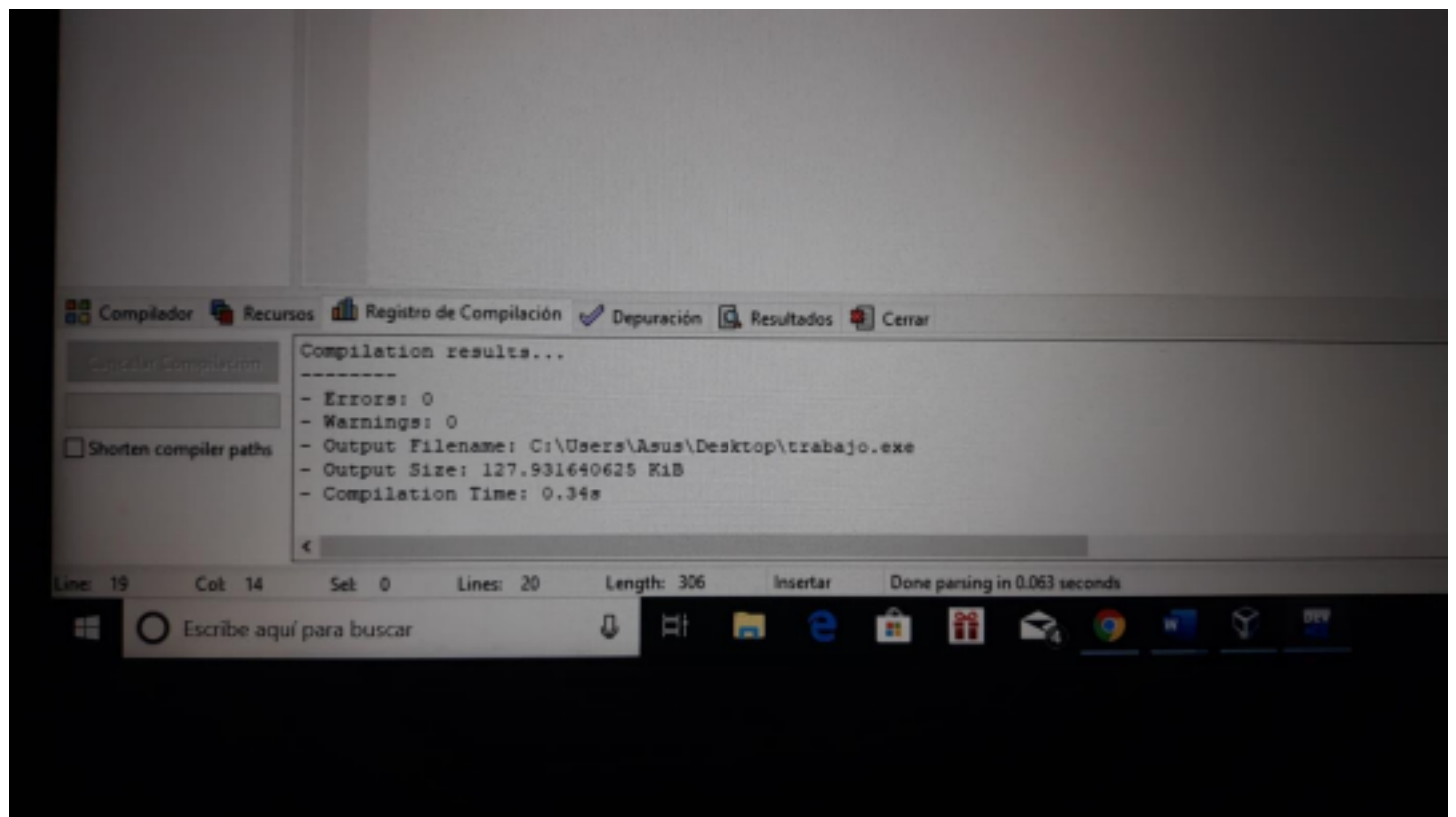
12 / 13

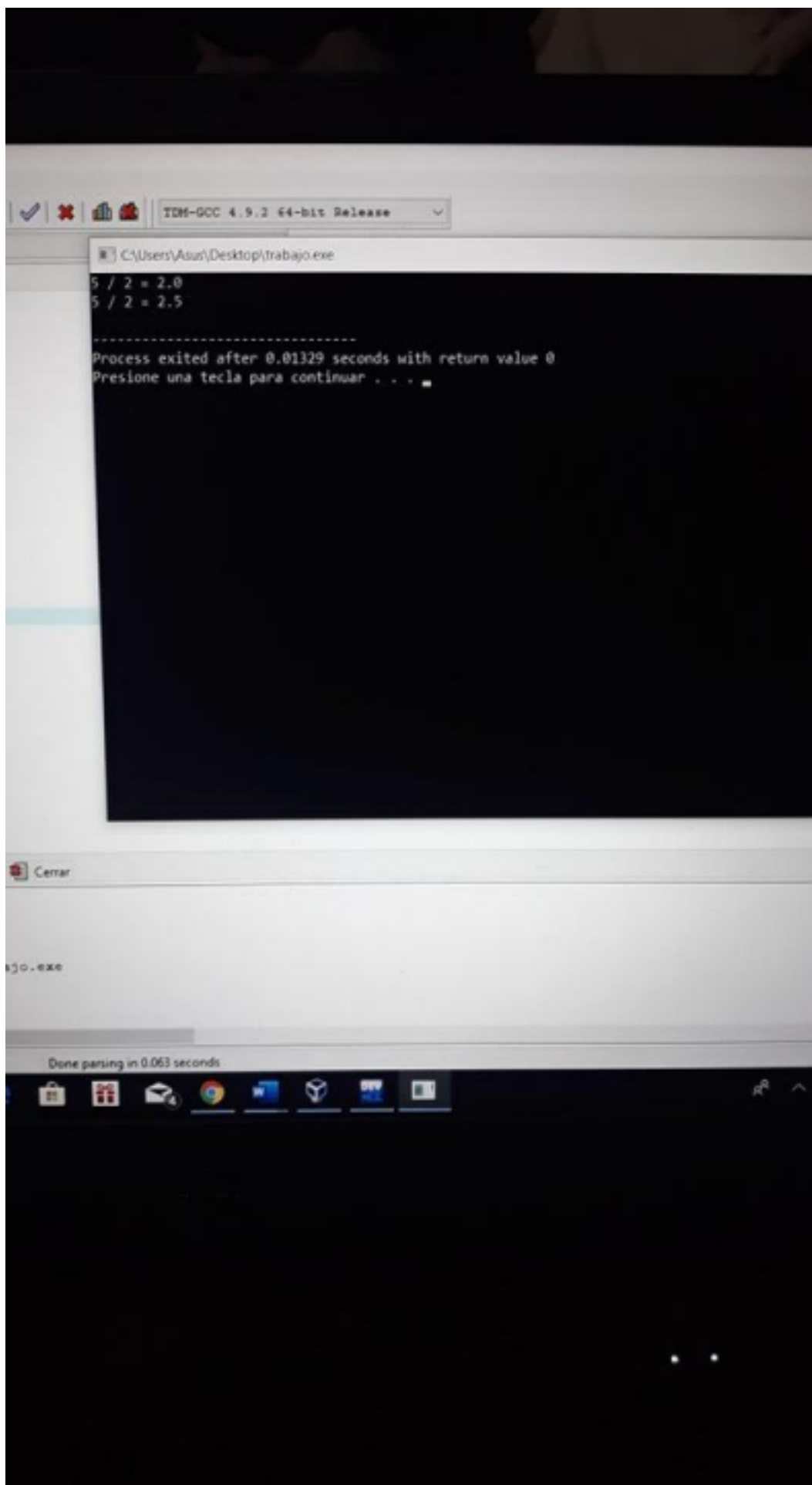
Página 12 de 12



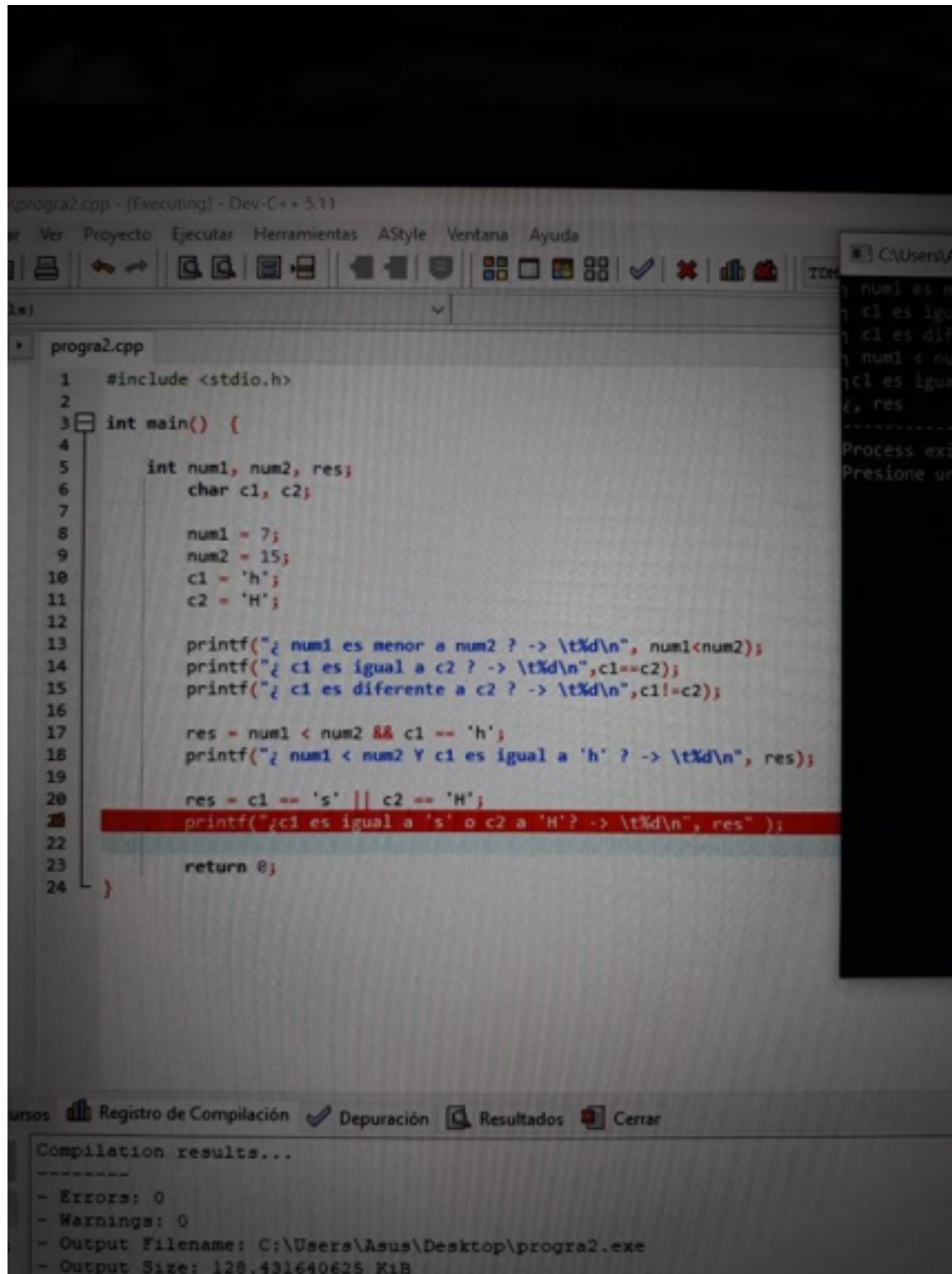








3._ Operadores lógicos







```
progra2.cpp - [Executing] - Dev-C++ 5.11
Ar Ver Proyecto Ejecutar Herramientas AStyle Ventana Ayuda
C:\Users\Asus\
num1 es me
c1 es igual
c1 es dife
num1 < num
c1 es igual
res
-----
Process exit
Presione una

1  #include <stdio.h>
2
3  int main() {
4
5      int num1, num2, res;
6      char c1, c2;
7
8      num1 = 7;
9      num2 = 15;
10     c1 = 'h';
11     c2 = 'H';
12
13     printf("¿ num1 es menor a num2 ? -> %d\n", num1 < num2);
14     printf("¿ c1 es igual a c2 ? -> %d\n", c1 == c2);
15     printf("¿ c1 es diferente a c2 ? -> %d\n", c1 != c2);
16
17     res = num1 < num2 && c1 == 'h';
18     printf("¿ num1 < num2 Y c1 es igual a 'h' ? -> %d\n", res);
19
20     res = c1 == 's' || c2 == 'H';
21     printf("¿ c1 es igual a 's' o c2 a 'H' ? -> %d\n", res);
22
23     return 0;
24 }
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Asus\Desktop\progra2.exe
- Output Size: 128.431640625 KiB

```
13     printf("¿ num1 es menor a num2 ? -> %d\n", num1<num2);
14     printf("¿ c1 es igual a c2 ? -> %d\n", c1==c2);
15     printf("¿ c1 es diferente a c2 ? -> %d\n", c1!=c2);
16
17     res = num1 < num2 && c1 == 'h';
18     printf("¿ num1 < num2 Y c1 es igual a 'h' ? -> %d\n", res);
19
20     res = c1 == 's' || c2 == 'H';
21     printf("¿c1 es igual a 's' o c2 a 'H'? -> %d\n", res);
22
23     return 0;
24 }
```

Recursos  Registro de Compilación  Depuración  Resultados  Cerrar

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Asus\Desktop\progra2.exe
- Output Size: 128.431640625 KiB
- Compilation Time: 0.55s

<

9 Sel: 0 Lines: 24 Length: 531 Insertar Done parsing in 0.016 seconds

be aquí para buscar



C:\Users\Asus\Desktop\progra2.exe

```
num1 es menor a num2 ? -> 1
c1 es igual a c2 ? -> 0
c1 es diferente a c2 ? -> 1
num1 < num2 Y c1 es igual a 'h' ? -> 1
c1 es igual a 's' o c2 a 'H'? -> -1938490832
¿, res
```

Process exited after 0.1461 seconds with return value 0

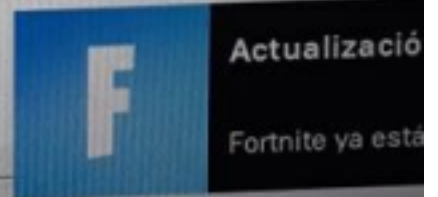
Presione una tecla para continuar . . .

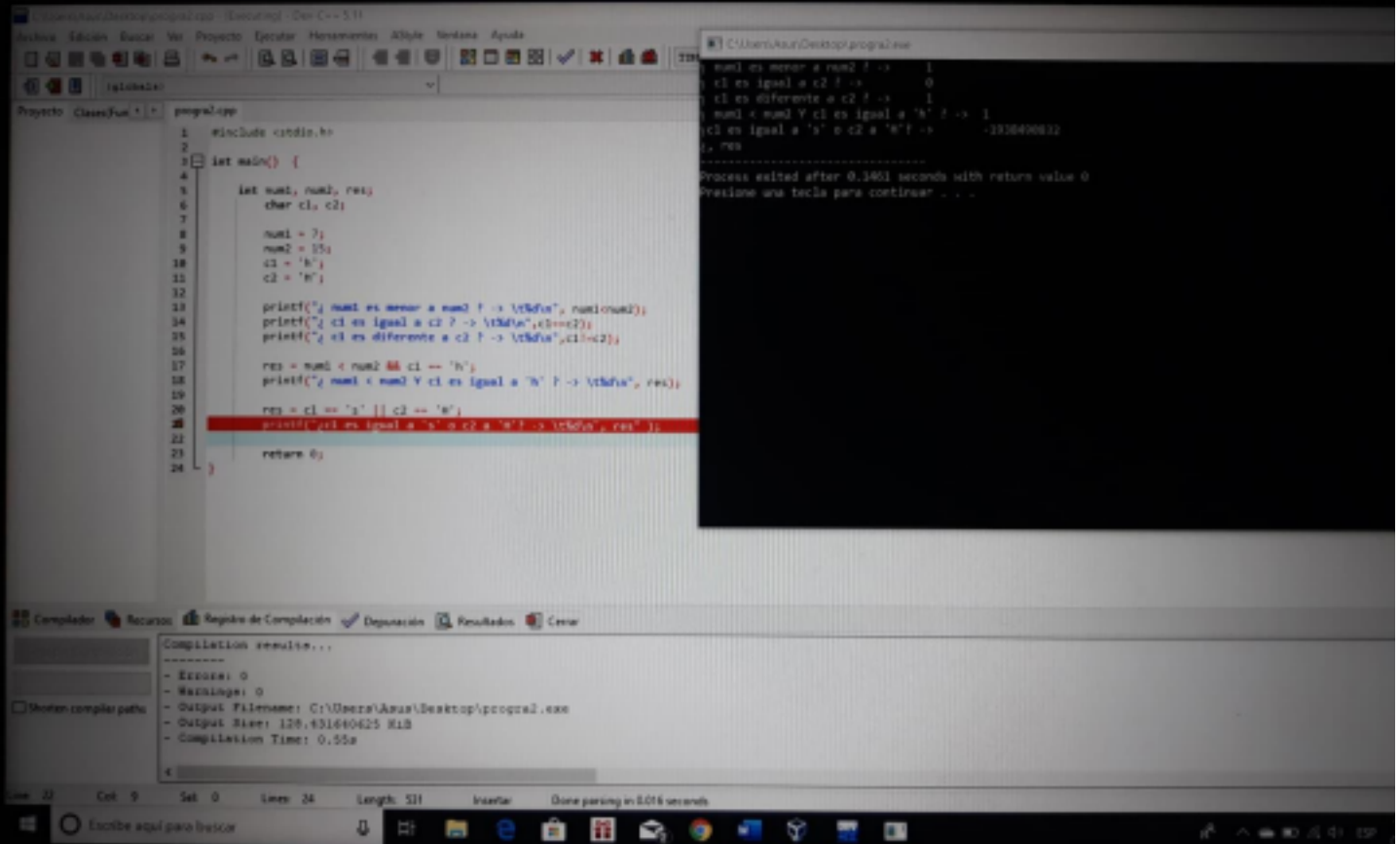
n2);

n", res);

es");

ing in 0.016 seconds





Conclusión:

Aprender los fundamentos de un lenguaje de programación es como abrir las puertas a la oportunidad y a la aventura. Y, en estos tiempos, aprender a programar, aún a un nivel básico, es bastante beneficioso, ya que el lenguaje C proporciona una base para la comprensión de los conceptos básicos de programación. Si sabe cómo programar en C, puede aprender C++, C#, Java, y muchos otros lenguajes de programación.