



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorios de docencia

Laboratorio de Computación Salas A y B

Profesor: GARCIA MORALES KARINA ING.

Asignatura: FUNDAMENTOS DE PROGRAMACION

Grupo: 1121

No de Práctica(s): 7

Integrante(s): JOSE DANIEL CALLEJAS SANDOVAL

*No. de Equipo de
cómputo empleado:* 25

Semestre: 1

Fecha de entrega: 09-10-2018

Observaciones:

CALIFICACIÓN: _____

PRACTICA 7: Fundamentos de Lenguaje C

OBJETIVOS: Elaborar programas en lenguaje C utilizando las instrucciones de control de tipo secuencia, para realizar la declaración de variables de diferentes tipos de datos, así como efectuar llamadas a funciones externas de entrada y salida para asignar y mostrar valores de variables y expresiones.

DESARROLLO DE LA PRACTICA:

Para crear un programa en C se siguen tres etapas principales: edición, compilación y ejecución.

Edición: Se escribe el código fuente en lenguaje C desde algún editor de textos.

Compilación: A partir del código fuente (lenguaje C) se genera el archivo en lenguaje máquina (se crea el programa objeto o ejecutable).

Ejecución: El archivo en lenguaje máquina se puede ejecutar en la arquitectura correspondiente.

Un programa en C consiste en una o más funciones, de las cuales una de ellas debe llamarse `main()` y es la principal.

Al momento de ejecutar un programa objeto (código binario), se ejecutarán únicamente las instrucciones que estén definidas dentro de la función principal. La función principal puede contener sentencias, estructuras de control y comentarios. Dentro de las sentencias se encuentran la declaración y/o asignación de variables, la realización de operaciones básicas, y las llamadas a funciones.

Comentarios: Es una buena práctica en cualquier lenguaje de programación realizar comentarios para documentar el programa. En C existen dos tipos de comentarios: el comentario por línea y el comentario por bloque.

El comentario por línea inicia cuando se insertan los símbolos `//'` y termina con el salto de línea (hasta donde termine el renglón).

El comentario por bloque inicia cuando se insertan los símbolos `/*` y termina cuando se encuentran los símbolos `*/`. Cabe resaltar que el comentario por bloque puede abarcar varios renglones.

Declaración de variables: Para declarar variables en C se sigue la siguiente sintaxis:

`[modificadores] tipoDeDato identificador [= valor];`

Por lo tanto, una variable puede tener modificadores (éstos se analizarán más adelante y son opcionales), debe declarar el tipo de dato que puede contener la variable, debe declarar el identificador (nombre o etiqueta) con el que se va a manejar el valor y se puede asignar un valor inicial a la variable (opcional).

También es posible declarar varios identificadores de un mismo tipo de dato e inicializarlos en el mismo renglón, lo único que se tiene que hacer es separar cada identificador por comas. `tipoDeDato identificador1 [= valor], identificador2 [= valor];` Tipos de datos Los tipos de datos básicos en C son:

Caracteres: codificación definida por la máquina.

Enteros: números sin punto decimal.

Flotantes: números reales de precisión normal.

Dobles: números reales de doble precisión.

Identificadores: Un identificador es el nombre con el que se va a almacenar en memoria un tipo de dato. Los identificadores siguen las siguientes reglas: Debe iniciar con una letra `[a-z]`.

Puede contener letras `[A-Z, a-z]`, números `[0-9]` y el carácter guion bajo `(_)`.

En la notación de camello los nombres de cada palabra empiezan con mayúscula y el resto se escribe con minúsculas (a excepción de la primera palabra, la cual inicia también con minúscula). No se usan puntos ni guiones para separar las palabras. Además, las palabras de las constantes se escriben con mayúsculas y se separan con guion bajo.

La biblioteca 'stdio.h' contiene diversas funciones tanto para imprimir en la salida estándar (monitor) como para leer de la entrada estándar (teclado).

Printf es una función para imprimir con formato, es decir, se tiene que especificar entre comillas el tipo de dato que se desea imprimir, también se puede combinar la impresión de un texto predeterminado.

Scanf es una función que sirve para leer datos de la entrada estándar (teclado), para ello únicamente se especifica el tipo de dato que se desea leer entre comillas y en qué variable se quiere almacenar. Al nombre de la variable le antecede un ampersand (&), esto indica que el dato recibido se guardará en la localidad de memoria asignada a esa variable.

Modificadores: Los modificadores que se pueden agregar al inicio de la declaración de variables son const y static.

El modificador const impide que una variable cambie su valor durante la ejecución del programa, es decir, permite para crear constantes. Por convención, las constantes se escriben con mayúsculas y se deben inicializar al momento de declararse.

Operadores

Los operadores aritméticos que maneja lenguaje C se describen a continuación:

+ Suma $125.78 + 62.5 = 188.28$

- Resta $65.3 - 32.33 = 32.97$

* Multiplicación $8.27 * 7 = 57.75$

/ División $15 / 4 = 3.75$

% Módulo $4 \% 2 = 0$

Expresiones lógicas

Las expresiones lógicas están constituidas por números, caracteres, constantes o variables que están relacionados entre sí por operadores lógicos. Una expresión lógica puede tomar únicamente los valores verdadero o falso.

Los operadores de relación permiten comparar elementos numéricos, alfanuméricos, constantes o variables.

Operador	Operación	Uso	Resultado
----------	-----------	-----	-----------

==	Igual que	'h' == 'H'	Falso
----	-----------	------------	-------

!=	Diferente a	'a' != 'b'	Verdadero
----	-------------	------------	-----------

<	Menor que	7 < 15	Verdadero
---	-----------	--------	-----------

>	Mayor que	11 > 22	Falso
---	-----------	---------	-------

<=	Menor o igual	15 <= 22	Verdadero
----	---------------	----------	-----------

>=	Mayor o igual	20 >= 35	Falso
----	---------------	----------	-------

Los operadores lógicos permiten formular condiciones complejas a partir de condiciones simples.

Depuración: Cuando un programa falla (no termina su ejecución de manera correcta) y la información enviada por el compilador es muy general, se puede ejecutar el programa en un contexto controlado para saber, exactamente, dónde está fallando.

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — vi programa.c — 80x24

```
#include <stdio.h>
int main ()
{
    int Entera=10;
    char Cara='B';
    float Flot=3.1415;
    double Dobl=315.64789;
    printf ("\n%d",Entera);
    printf ("\n%c",Cara);
    printf ("\n%f",Flot);
    printf ("\n%lf",Dobl);
    //mi primer programa
}
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — -bash — 80x24

```
fp1121alu06:~$ vi programa.c
Bosnia18:~$ fp1121alu06$ gcc programa.c
Bosnia18:~$ fp1121alu06$ ./programa.out
10
B
3.141500
315.647890
Bosnia18:~$ fp1121alu06$
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — vi programa.c — 80x24

```
#include <stdio.h>
int main ()
{
    int Entera=10;
    char Cara='B';
    float Flot=3.1415;
    double Dobl=315.64789;
    printf ("\n %d",Entera);
    printf ("\n %c",Cara);
    printf ("\n %f",Flot);
    printf ("\n %lf",Dobl);
    return 0;
}
//mi primer programa
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — -bash — 80x24

```
Bosnia18:~$ fp1121alu06$ vi programa.c
Bosnia18:~$ fp1121alu06$ gcc programa.c -o programa.out
Bosnia18:~$ fp1121alu06$ ./programa.out
10
B
3.141500
315.647890
Bosnia18:~$ fp1121alu06$
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — -bash — 80x24

```
Bosnia18:~$ fp1121alu06$ vi programa.c
Bosnia18:~$ fp1121alu06$ gcc programa.c -o programa.out
Bosnia18:~$ fp1121alu06$ ./programa.out
10
B
3.141500
315.647890
Bosnia18:~$ fp1121alu06$
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — vi pScanf.c — 80x24

```
#include <stdio.h>
/*
Este programa muestra la manera en la que se declaran y asignan variables de dif
erentes tipos: numéricas (enteras y reales) y caracteres, así como la manera en
la que se imprimen los diferentes tipos de datos.
*/
int main() {
    /* Es recomendable al inicio declarar
    todas las variables que se van a utilizar
    en el programa */ // variables enteras
    int enteroNumero;
    char caracterA = 65;
    // Variable reales
    double puntoFlotanteNumero;
    // Convierte el entero a carácter (ASCII)
    // Asignar un valor del teclado a una variable
    printf("Escriba un valor entero: ");
    scanf("%d", &enteroNumero);
    printf("Escriba un valor real: ");
    scanf("%lf", &puntoFlotanteNumero);
    // Imprimir los valores con formato
    printf("\nImprimiendo las variables enteras:\n");
    printf("\tValor de enteroNumero = %i\n", enteroNumero);
    printf("\tValor de caracterA = %c\n", puntoFlotanteNumero);
    printf("\tValor de enteroNumero en base 16 = %x\n", enteroNumero);
    printf("\tValor de caracterA en código hexadecimal = %x\n", puntoFlotanteNumero);
    printf("\tValor de puntoFlotanteNumero en notación científica = %e\n", puntoFlot
anteNumero);
    // La función getchar() espera un carácter para continuar la ejecución getchar()
    return 0;
}
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — -bash — 80x24

```
Bosnia18:~$ fp1121alu06$ vi TAREA1.c
Bosnia18:~$ fp1121alu06$ gcc TAREA1.c -o TAREA1.out
TAREA1.c:21: warning: type specifier missing, defaults to 'int'
[...Wimplicit-int...]
main()
{
    1 warning generated.
Bosnia18:~$ fp1121alu06$ vi TAREA1.c
Bosnia18:~$ fp1121alu06$ gcc pScanf.c -o pScanf.out
Bosnia18:~$ fp1121alu06$ ./pScanf.c
Bosnia18:~$ fp1121alu06$ ./pScanf.c
Bosnia18:~$ fp1121alu06$ ./pScanf.c
Escriba un valor entero: 15
Escriba un valor real: 20

Imprimiendo las variables enteras:
Valor de enteroNumero = 15
Valor de caracterA = A
Valor de puntoFlotanteNumero = 20.000000

Valor de enteroNumero en base 16 = f
Valor de caracterA en código hexadecimal = 15
Valor de puntoFlotanteNumero en notación científica = 2.000000e+01
Bosnia18:~$ fp1121alu06$
```

TerminalShellEdiciónVisualizaciónVentanaAyuda

fp1121alu06 — -bash — 80x24

```
Bosnia18:~$ fp1121alu06$ vi TAREA1.c
Bosnia18:~$ fp1121alu06$ gcc TAREA1.c -o TAREA1.out
TAREA1.c:21: warning: type specifier missing, defaults to 'int'
[...Wimplicit-int...]
main()
{
    1 warning generated.
Bosnia18:~$ fp1121alu06$ vi TAREA1.c
Bosnia18:~$ fp1121alu06$ gcc pScanf.c -o pScanf.out
Bosnia18:~$ fp1121alu06$ ./pScanf.c
Bosnia18:~$ fp1121alu06$ ./pScanf.c
Bosnia18:~$ fp1121alu06$ ./pScanf.c
Escriba un valor entero: 15
Escriba un valor real: 20

Imprimiendo las variables enteras:
Valor de enteroNumero = 15
Valor de caracterA = A
Valor de puntoFlotanteNumero = 20.000000

Valor de enteroNumero en base 16 = f
Valor de caracterA en código hexadecimal = 15
Valor de puntoFlotanteNumero en notación científica = 2.000000e+01
Bosnia18:~$ fp1121alu06$
```

EJERCICIOS DE TAREA:

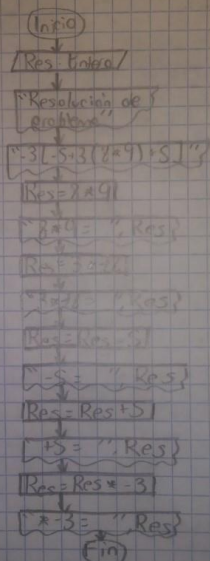
ANALISIS Y DIAGRAMAS DE FLUJO:

PROGRAMA1

Analisis: DE: No hay

Res: Ninguna

DS: Operacion

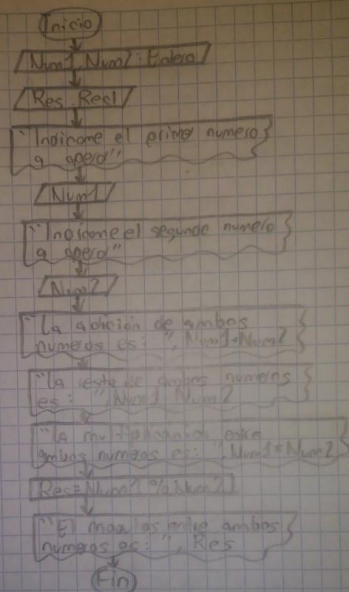


PROGRAMA3

Analisis: DE: 2 numeros y oper

Res: Ninguna

DS: 4 operaciones

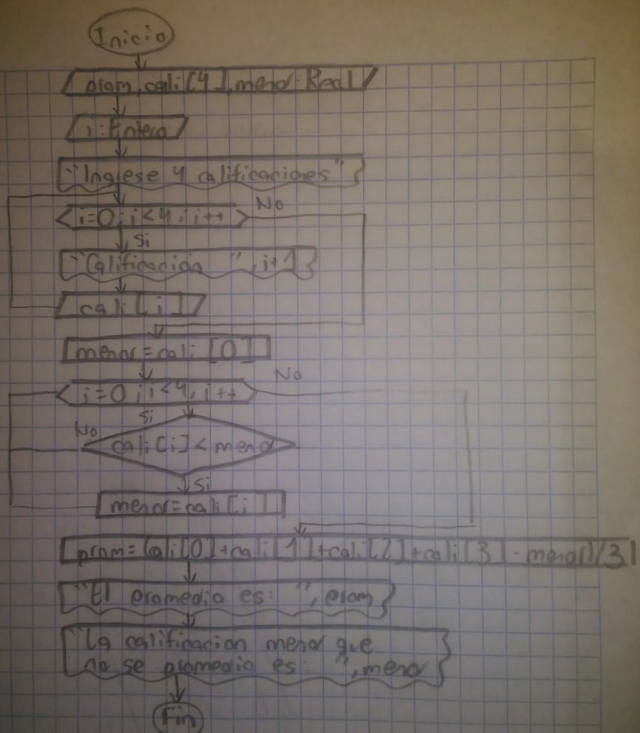


PROGRAMA4

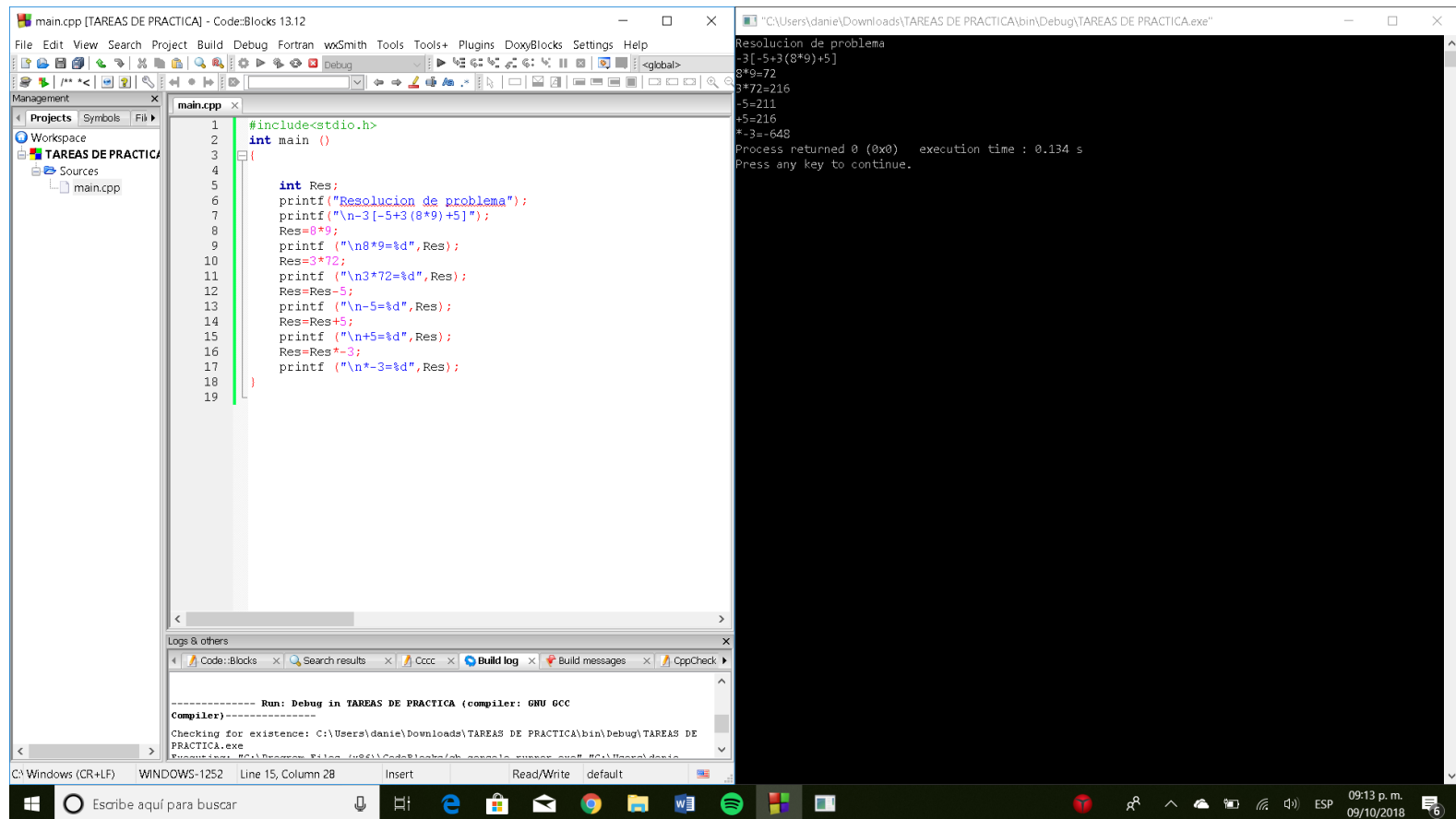
Analisis: DE: 4 calificaciones

Res: Ninguna

DS: Promedio



PROGRAMA 1



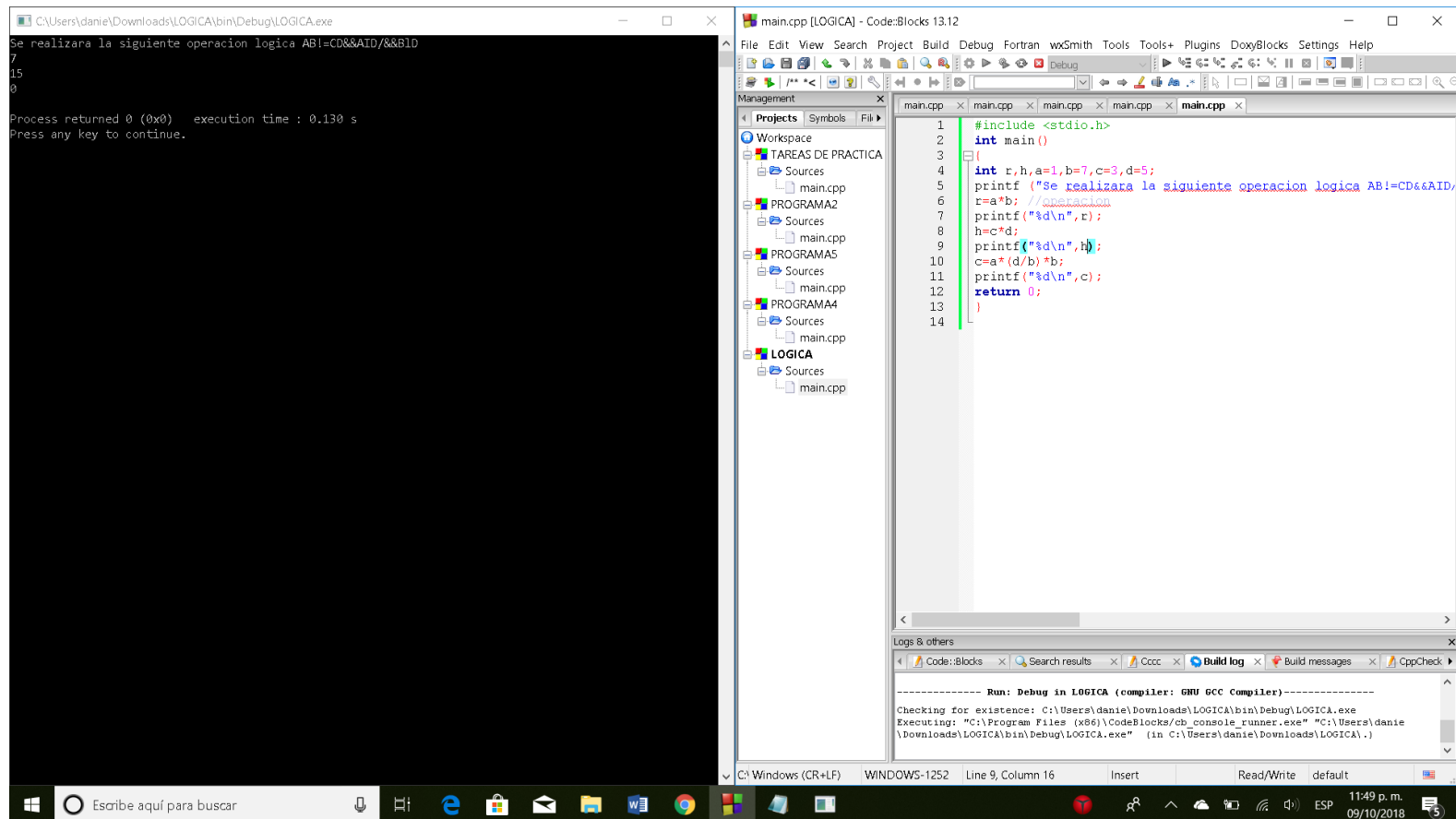
```
main.cpp [TAREAS DE PRACTICA] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Workspace
TAREAS DE PRACTICA
Sources
main.cpp
1 #include<stdio.h>
2 int main ()
3 {
4
5     int Res;
6     printf("Resolucion de problema");
7     printf("\n-5+3(8*9)+5");
8     Res=0*9;
9     printf ("\n8*9=%d",Res);
10    Res=3*72;
11    printf ("\n3*72=%d",Res);
12    Res=Res-5;
13    printf ("\n-5=%d",Res);
14    Res=Res+5;
15    printf ("\n+5=%d",Res);
16    Res=Res*-3;
17    printf ("\n*-3=%d",Res);
18 }
19

Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck

----- Run: Debug in TAREAS DE PRACTICA (compiler: GNU GCC Compiler)-----
Checking for existence: C:\Users\danie\Downloads\TAREAS DE PRACTICA\bin\Debug\TAREAS DE PRACTICA.exe
Executing: "C:\Program Files (x86)\CodeBlocks\cb_console_runner.exe" "C:\Users\danie\Downloads\TAREAS DE PRACTICA\bin\Debug\TAREAS DE PRACTICA.exe"

Resolucion de problema
-5+3(8*9)+5
8*9=72
3*72=216
-5=211
+5=216
*-3=-648
Process returned 0 (0x0) execution time : 0.134 s
Press any key to continue.
```

PROGRAMA 2



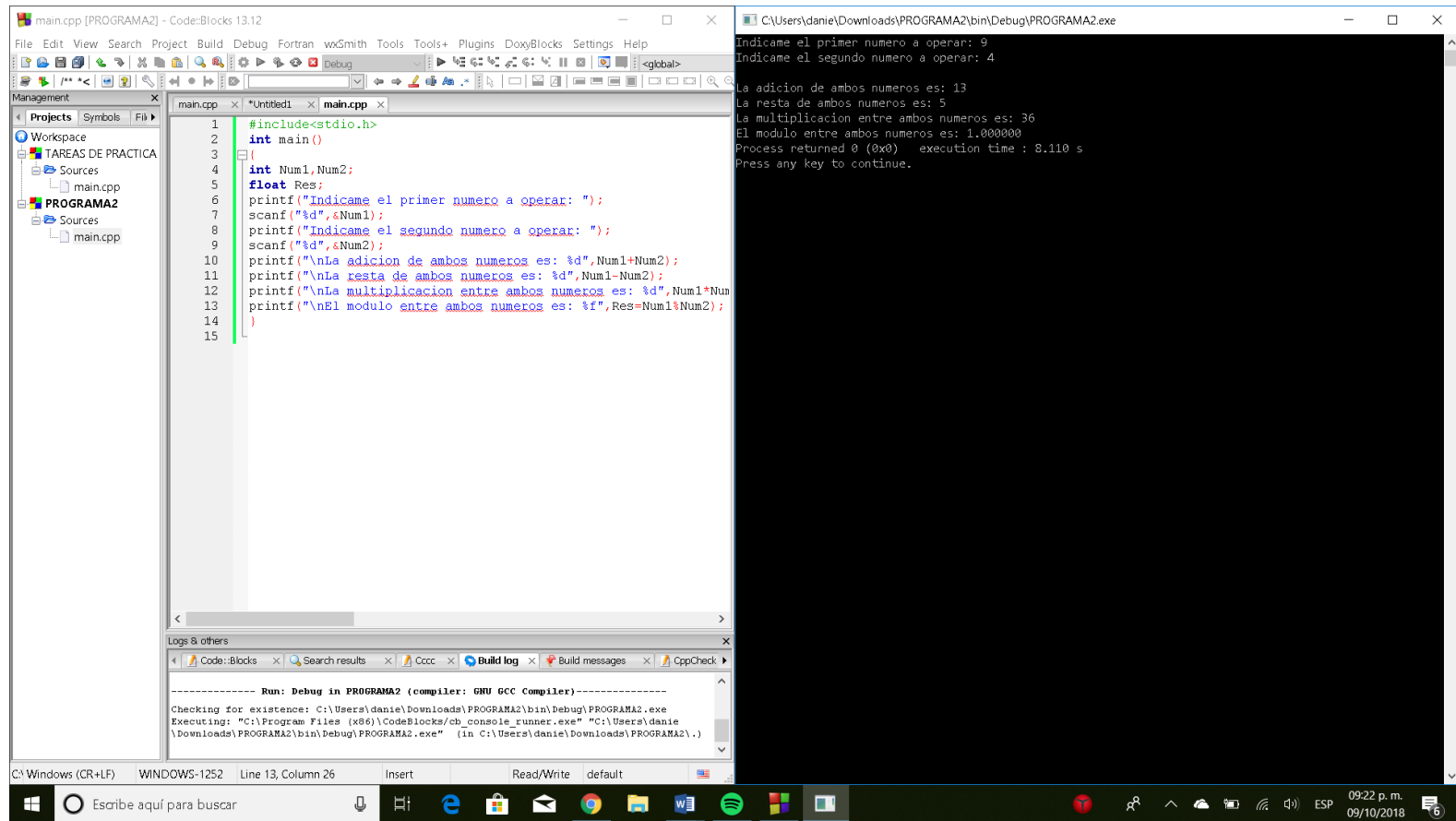
```
C:\Users\danie\Downloads\LOGICA\bin\Debug\LOGICA.exe
Se realizara la siguiente operacion logica AB!=CD&&AID/8&B1D
7
15
0
Process returned 0 (0x0) execution time : 0.130 s
Press any key to continue.

main.cpp [LOGICA] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Workspace
TAREAS DE PRACTICA
Sources
main.cpp
PROGRAMA2
Sources
main.cpp
PROGRAMAS
Sources
main.cpp
PROGRAMA4
Sources
main.cpp
LOGICA
Sources
main.cpp
1 #include <stdio.h>
2 int main()
3 {
4     int r,h,a=1,b=7,c=3,d=5;
5     printf ("Se realizara la siguiente operacion logica AB!=CD&&AID/8&B1D");
6     r=a*b; //operacion logica
7     printf ("%d\n",r);
8     h=c*d;
9     printf ("%d\n",h);
10    c=a*(d/b)*b;
11    printf ("%d\n",c);
12    return 0;
13 }
14

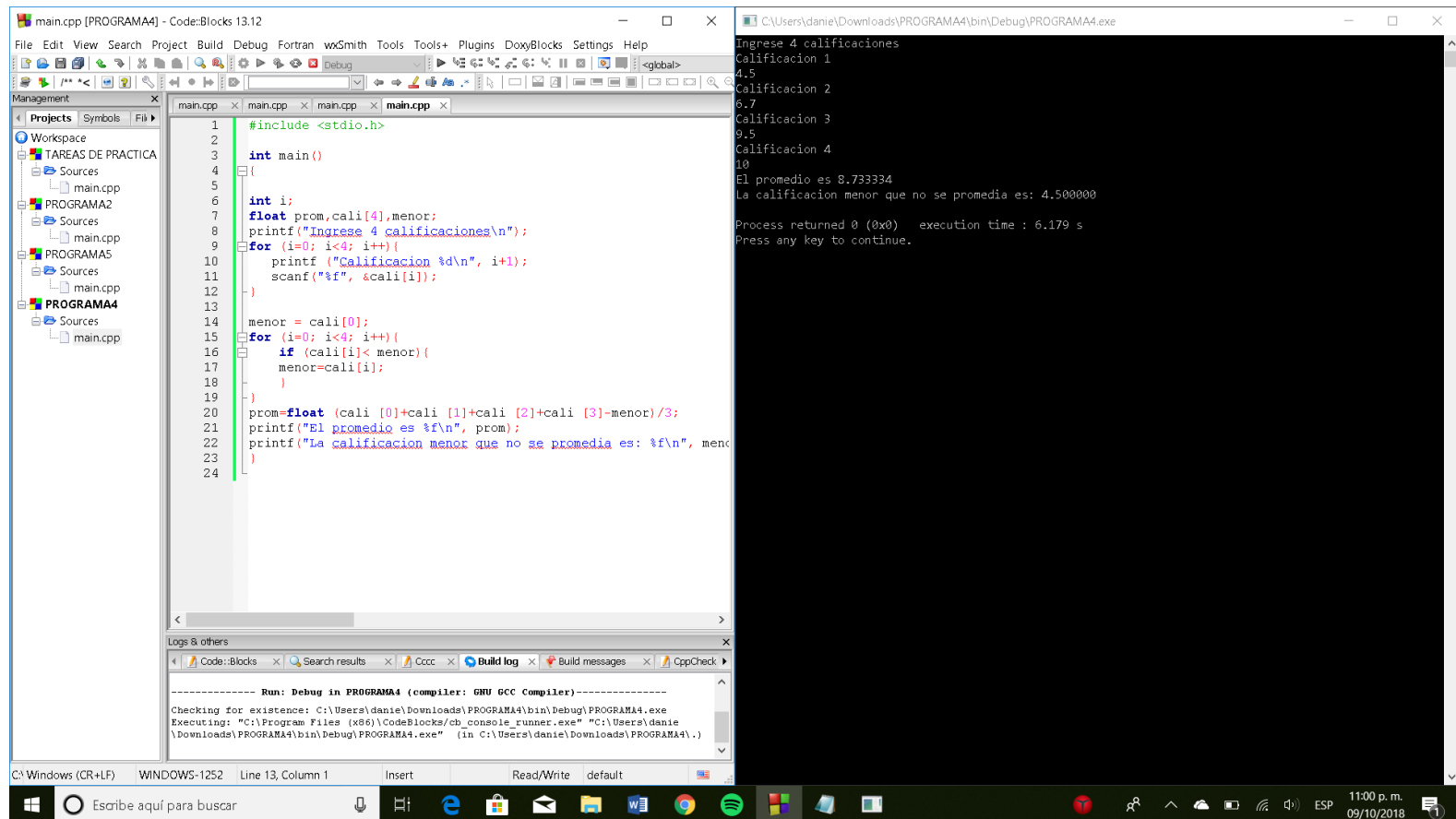
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck

----- Run: Debug in LOGICA (compiler: GNU GCC Compiler)-----
Checking for existence: C:\Users\danie\Downloads\LOGICA\bin\Debug\LOGICA.exe
Executing: "C:\Program Files (x86)\CodeBlocks\cb_console_runner.exe" "C:\Users\danie\Downloads\LOGICA\bin\Debug\LOGICA.exe" (in C:\Users\danie\Downloads\LOGICA\.)
```

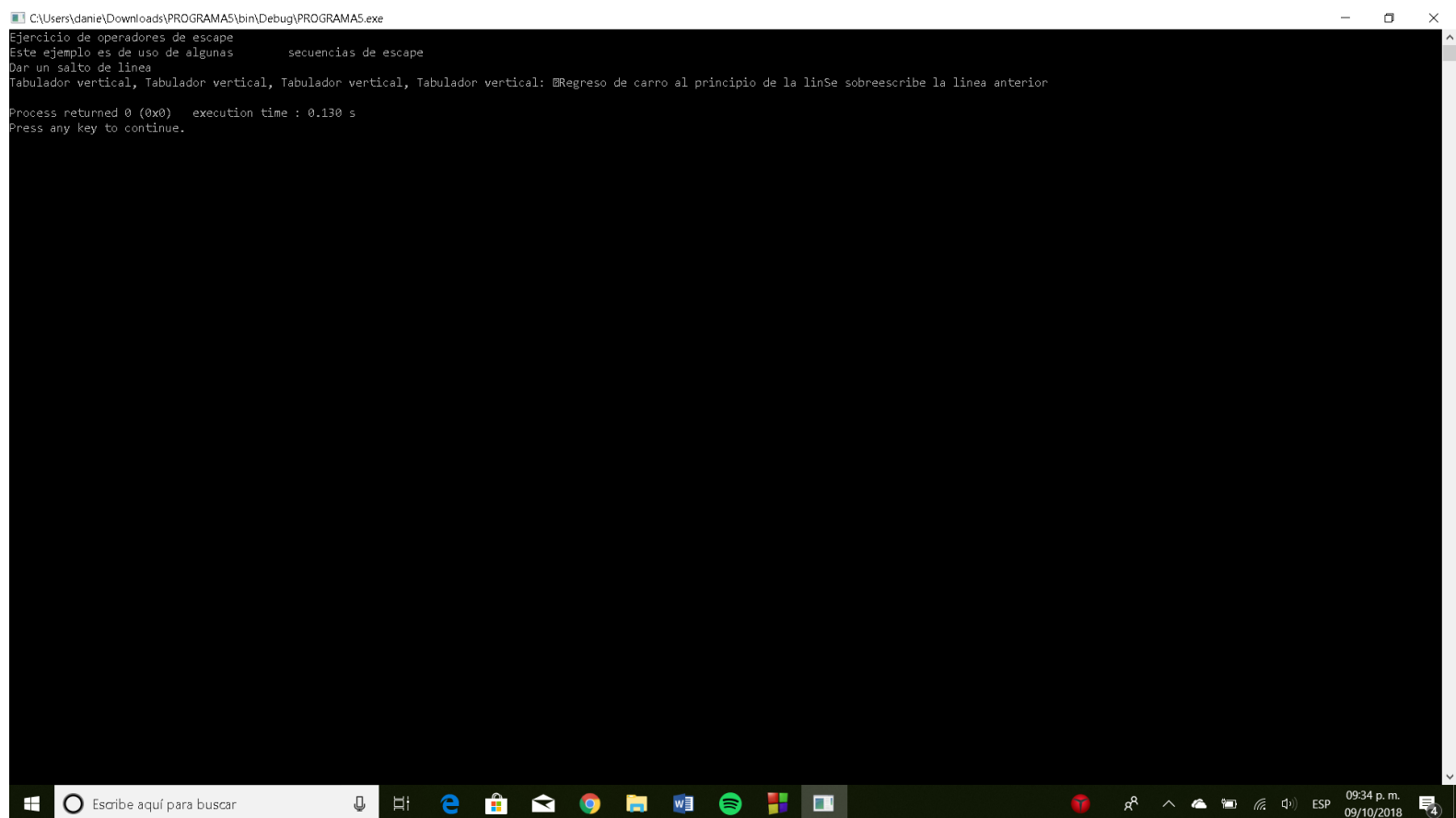
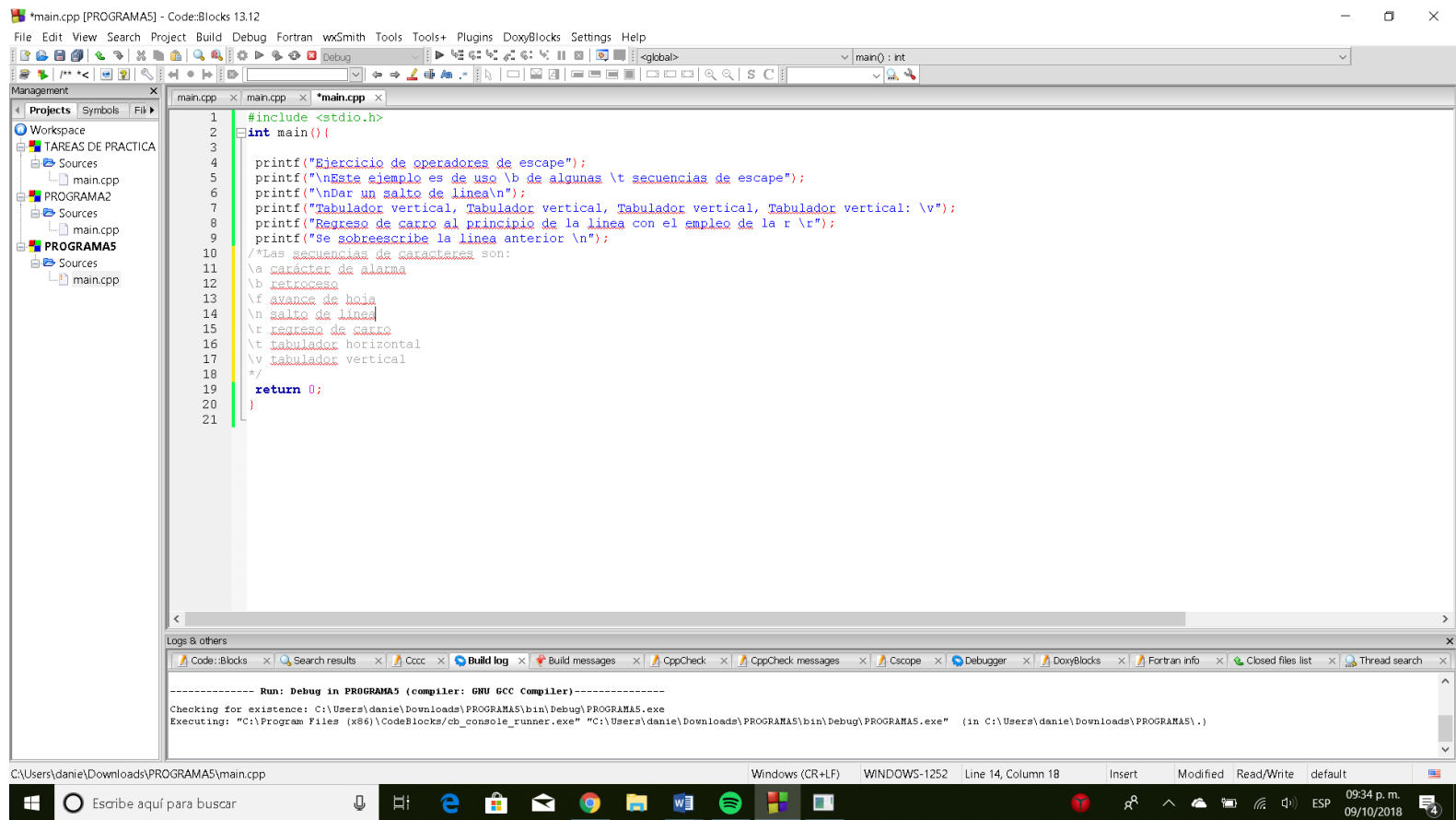
PROGRAMA 3



PROGRAMA 4



PROGRAMA 5



CONCLUSIONES: Al finalizar práctica, comprendí de mejor manera como se utilizan y como funcionan las Funciones, para poder simplificar el trabajo y optimizarlo.

De lo mas importante, fueron las secuencias de escape, y como estas no ayudan a darle presentación al programa.

La clase tuvo un ritmo rápido, aunque cada distinto comando o característica la vimos en detenimiento y profundidad, Los ejercicios de tarea me ayudaron a mejorar conocimientos previos y a optimizar programas.

BIBLIOGRAFIA:

- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.
- Carlos Guadalupe (2013). Aseguramiento de la calidad del software (SQA). [Figura 1]. Consulta: junio de 2015. Disponible en: <https://www.mindmeister.com/es/273953719/aseguramiento-de-la-calidad-delsoftware-sqa>