

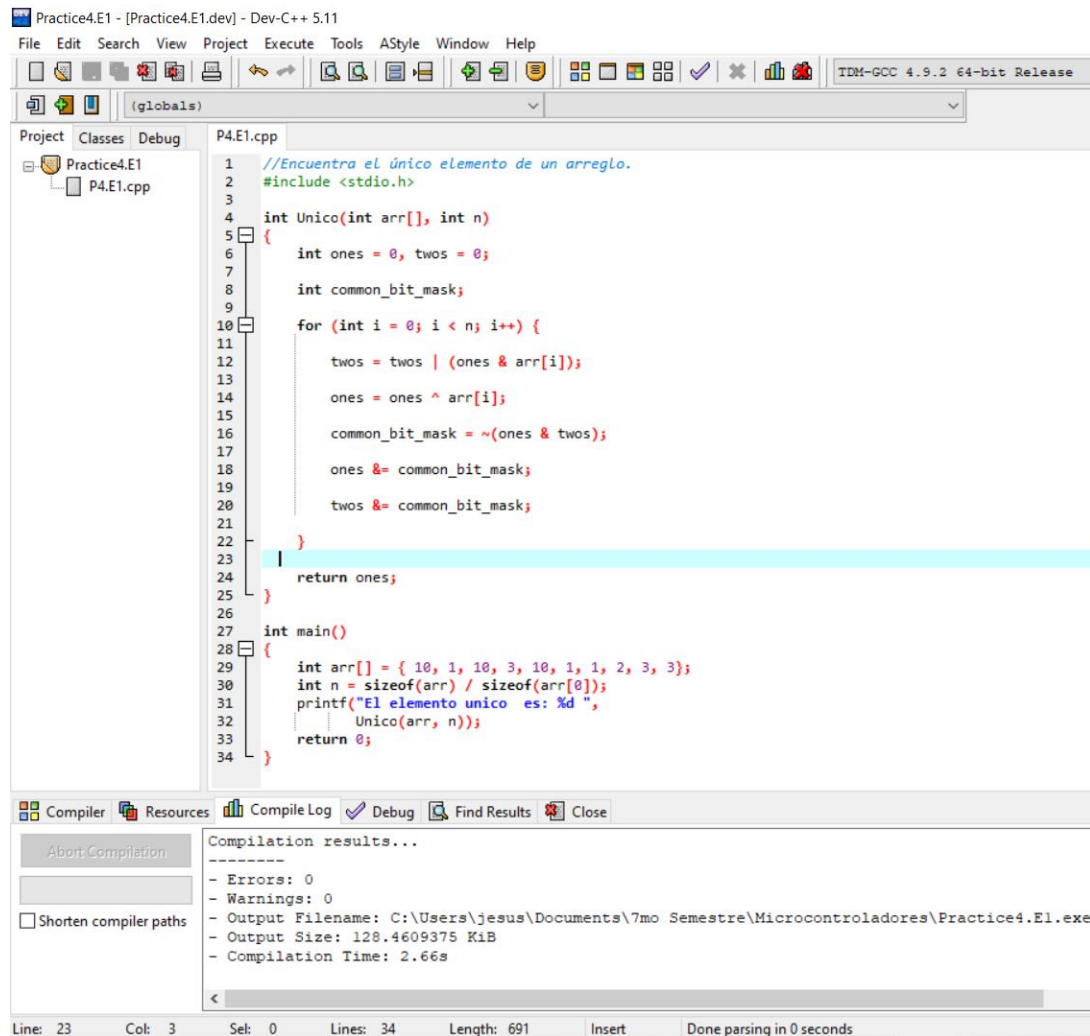
Jesus Ramiro Garza Hernandez

A01410925

Reporte Práctica 4

Ejercicio 1.- Encuentra el único elemento de un arreglo.

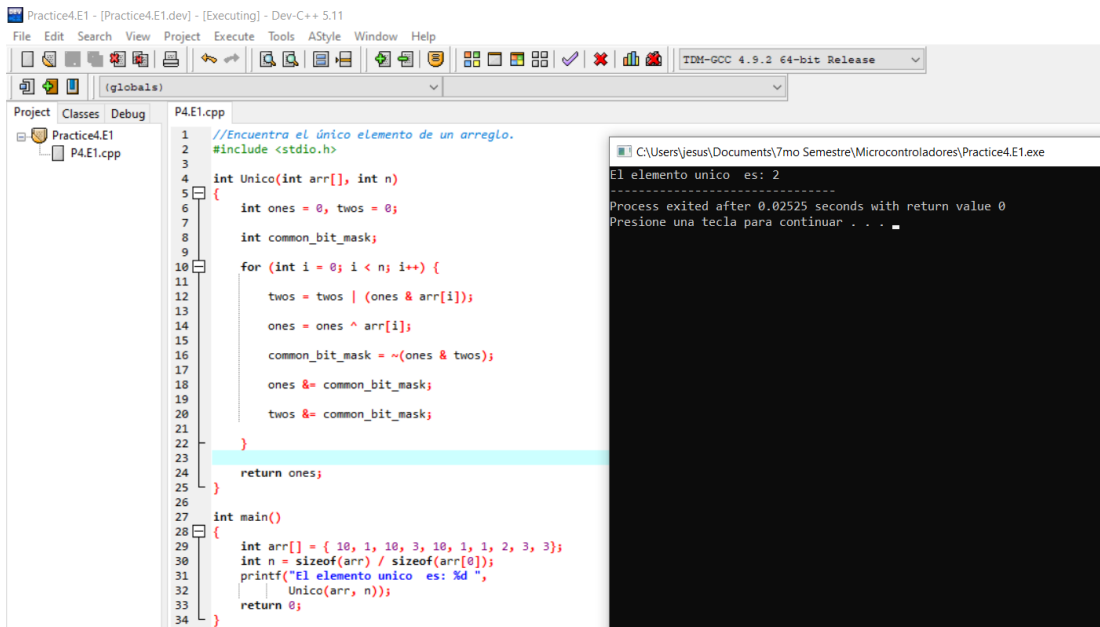
Código:



```
Practice4.E1 - [Practice4.E1.dev] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug
Practice4.E1
  P4.E1.cpp
1 //Encuentra el único elemento de un arreglo.
2 #include <stdio.h>
3
4 int Unico(int arr[], int n)
5 {
6     int ones = 0, twos = 0;
7
8     int common_bit_mask;
9
10    for (int i = 0; i < n; i++) {
11
12        twos = twos | (ones & arr[i]);
13
14        ones = ones ^ arr[i];
15
16        common_bit_mask = ~(ones & twos);
17
18        ones &= common_bit_mask;
19
20        twos &= common_bit_mask;
21
22    }
23
24    return ones;
25 }
26
27 int main()
28 {
29     int arr[] = { 10, 1, 10, 3, 10, 1, 1, 2, 3, 3};
30     int n = sizeof(arr) / sizeof(arr[0]);
31     printf("El elemento unico es: %d ",
32           Unico(arr, n));
33     return 0;
34 }
Compiler Resources Compile Log Debug Find Results Close
Compilation results...
-----
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.E1.exe
- Output Size: 128.4609375 KiB
- Compilation Time: 2.66s
Line: 23 Col: 3 Sel: 0 Lines: 34 Length: 691 Insert Done parsing in 0 seconds
```

Como se puede ver en el código, el array para analizar se establece en la variable `arr[]` y utilizando funciones lógicas como and-s y or-s se logra hacer la búsqueda del elemento único dentro del array. Para al final ser llamada y mostrada en pantalla.

Funcionamiento:



Practice4.E1 - [Practice4.E1.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug

Practice4.E1

P4.E1.cpp

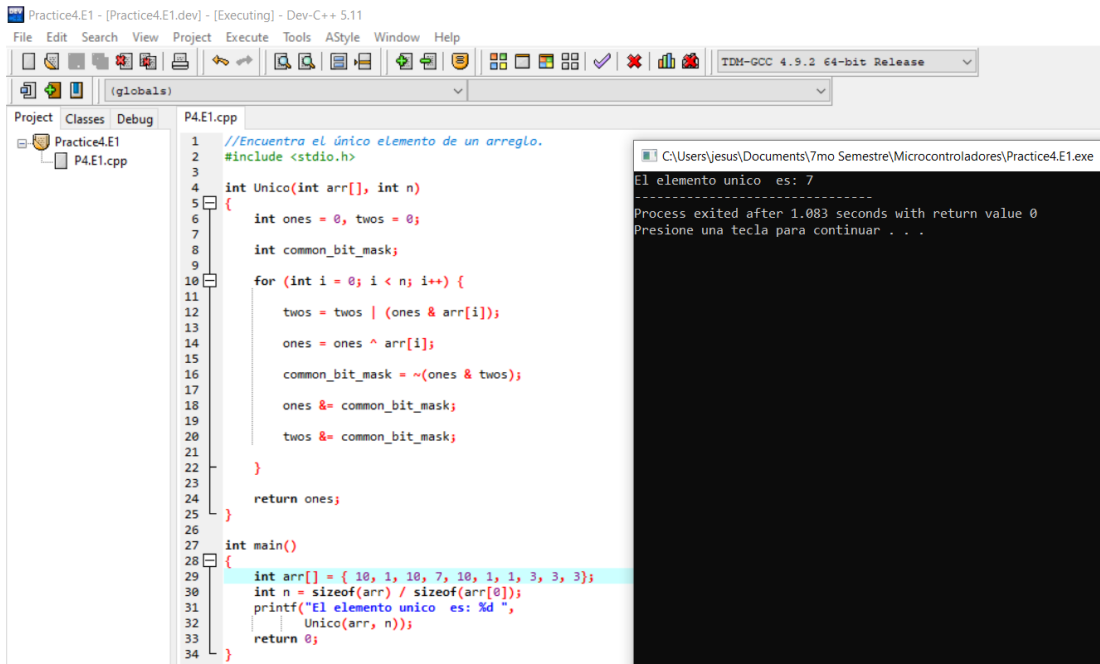
```
1 //Encuentra el único elemento de un arreglo.
2 #include <stdio.h>
3
4 int Unico(int arr[], int n)
5 {
6     int ones = 0, twos = 0;
7
8     int common_bit_mask;
9
10    for (int i = 0; i < n; i++) {
11        twos = twos | (ones & arr[i]);
12        ones = ones ^ arr[i];
13        common_bit_mask = ~(ones & twos);
14        ones &= common_bit_mask;
15        twos &= common_bit_mask;
16    }
17    return ones;
18 }
19
20 int main()
21 {
22     int arr[] = { 10, 1, 10, 3, 10, 1, 1, 2, 3, 3};
23     int n = sizeof(arr) / sizeof(arr[0]);
24     printf("El elemento unico es: %d ",
25           Unico(arr, n));
26     return 0;
27 }
```

C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.E1.exe

El elemento unico es: 2

Process exited after 0.02525 seconds with return value 0

Presione una tecla para continuar . . .



Practice4.E1 - [Practice4.E1.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug

Practice4.E1

P4.E1.cpp

```
1 //Encuentra el único elemento de un arreglo.
2 #include <stdio.h>
3
4 int Unico(int arr[], int n)
5 {
6     int ones = 0, twos = 0;
7
8     int common_bit_mask;
9
10    for (int i = 0; i < n; i++) {
11        twos = twos | (ones & arr[i]);
12        ones = ones ^ arr[i];
13        common_bit_mask = ~(ones & twos);
14        ones &= common_bit_mask;
15        twos &= common_bit_mask;
16    }
17    return ones;
18 }
19
20 int main()
21 {
22     int arr[] = { 10, 1, 10, 7, 10, 1, 1, 3, 3, 3};
23     int n = sizeof(arr) / sizeof(arr[0]);
24     printf("El elemento unico es: %d ",
25           Unico(arr, n));
26     return 0;
27 }
```

C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.E1.exe

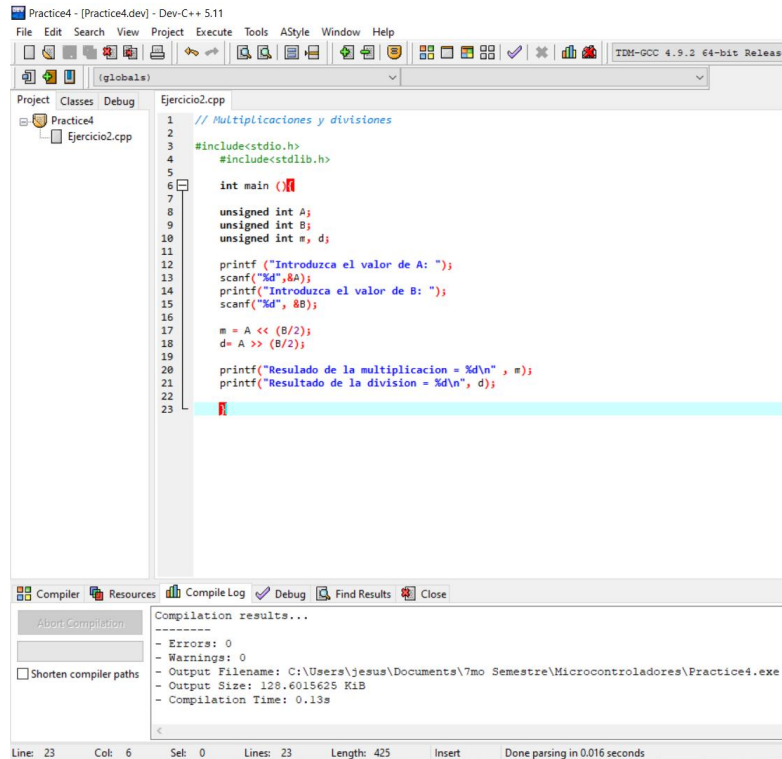
El elemento unico es: 7

Process exited after 1.083 seconds with return value 0

Presione una tecla para continuar . . .

Ejercicio 2.- Multiplicaciones y divisiones

Código:



```
1 // Multiplicaciones y divisiones
2
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 int main ()
7 {
8     unsigned int A;
9     unsigned int B;
10    unsigned int m, d;
11
12    printf ("Introduzca el valor de A: ");
13    scanf ("%d",&A);
14    printf ("Introduzca el valor de B: ");
15    scanf ("%d",&B);
16
17    m = A << (B/2);
18    d = A >> (B/2);
19
20    printf ("Resultado de la multiplicacion = %d\n", m);
21    printf ("Resultado de la division = %d\n", d);
22
23 }
```

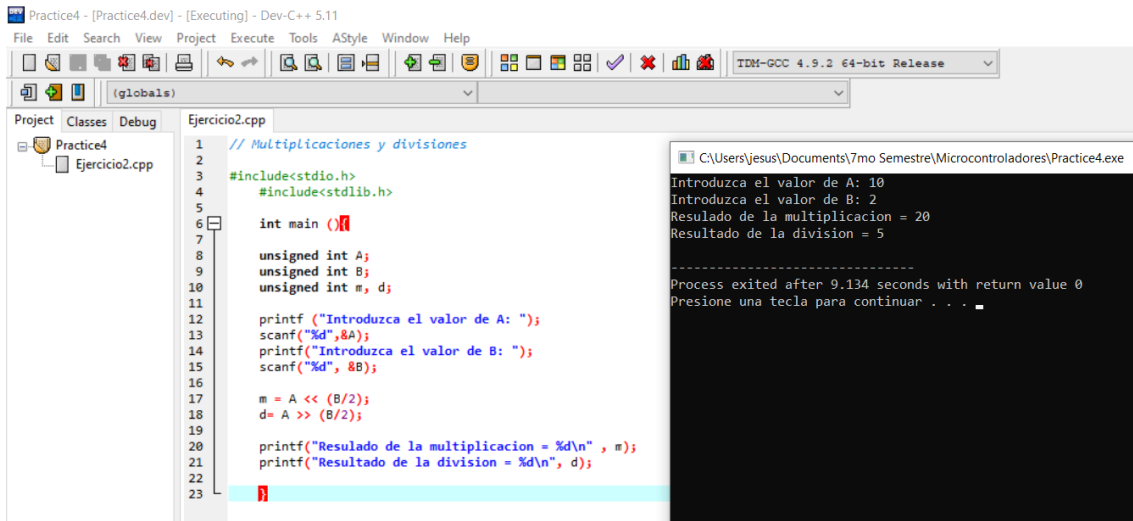
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.exe
- Output Size: 128.6015625 KiB
- Compilation Time: 0.13s

Line: 23 Col: 6 Sel: 0 Lines: 23 Length: 425 Insert Done parsing in 0.016 seconds

Para lograr la multiplicación de dos números decimales en operaciones de bit se utilizan los operadores << y para las divisiones se utilizan los operadores >> sin embargo debido a como se manipulan los bits es necesario que el numero introducido en B esté siendo dividido en la operación para que funcione correctamente.

Funcionamiento:



Practice4 - [Practice4.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

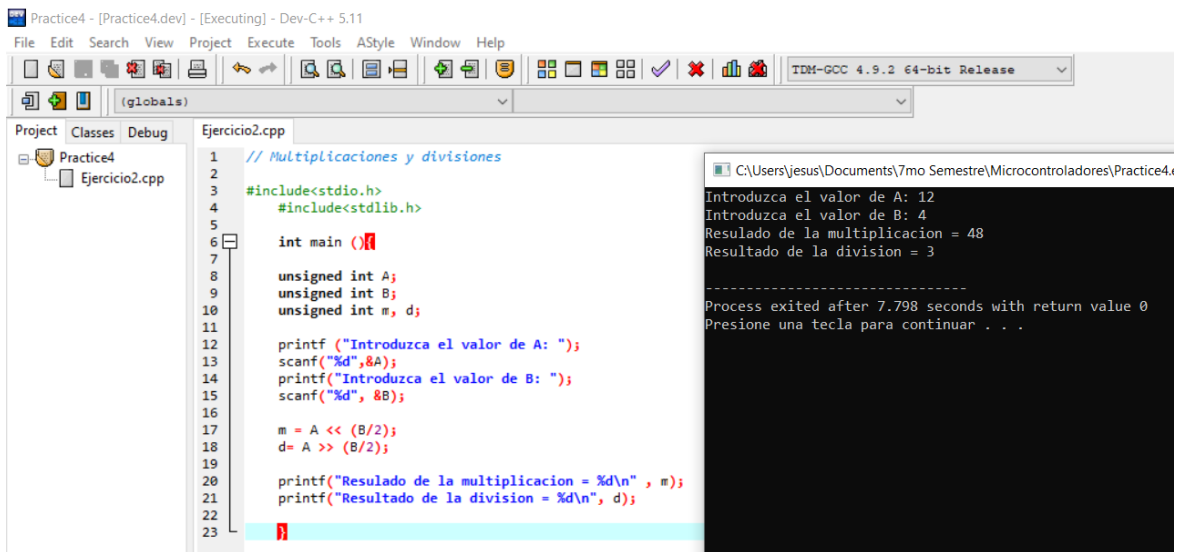
Project Classes Debug Ejercicio2.cpp

```
1 // Multiplicaciones y divisiones
2
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 int main ()
7 {
8     unsigned int A;
9     unsigned int B;
10    unsigned int m, d;
11
12    printf ("Introduzca el valor de A: ");
13    scanf ("%d",&A);
14    printf ("Introduzca el valor de B: ");
15    scanf ("%d", &B);
16
17    m = A << (B/2);
18    d= A >> (B/2);
19
20    printf ("Resultado de la multiplicacion = %d\n", m);
21    printf ("Resultado de la division = %d\n", d);
22
23 }
```

C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.exe

Introduzca el valor de A: 10
Introduzca el valor de B: 2
Resultado de la multiplicacion = 20
Resultado de la division = 5

Process exited after 9.134 seconds with return value 0
Presione una tecla para continuar . . .



Practice4 - [Practice4.dev] - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

(globals)

Project Classes Debug Ejercicio2.cpp

```
1 // Multiplicaciones y divisiones
2
3 #include<stdio.h>
4 #include<stdlib.h>
5
6 int main ()
7 {
8     unsigned int A;
9     unsigned int B;
10    unsigned int m, d;
11
12    printf ("Introduzca el valor de A: ");
13    scanf ("%d",&A);
14    printf ("Introduzca el valor de B: ");
15    scanf ("%d", &B);
16
17    m = A << (B/2);
18    d= A >> (B/2);
19
20    printf ("Resultado de la multiplicacion = %d\n", m);
21    printf ("Resultado de la division = %d\n", d);
22
23 }
```

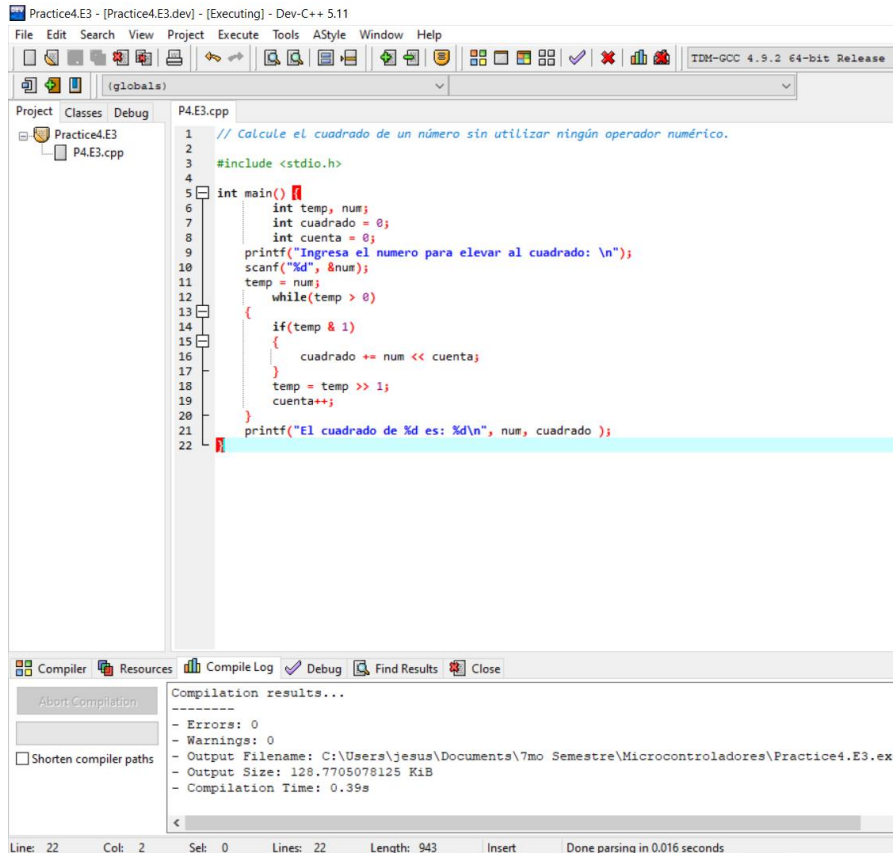
C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.exe

Introduzca el valor de A: 12
Introduzca el valor de B: 4
Resultado de la multiplicacion = 48
Resultado de la division = 3

Process exited after 7.798 seconds with return value 0
Presione una tecla para continuar . . .

Ejercicio 3.- Calcule el cuadrado de un número sin utilizar ningún operador numérico.

Código:



```
Practice4.E3 - [Practice4.E3.dev] - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
(globals)
Project Classes Debug P4.E3.cpp
Practice4.E3
P4.E3.cpp
1 // Calcule el cuadrado de un número sin utilizar ningún operador numérico.
2
3 #include <stdio.h>
4
5 int main()
6 {
7     int temp, num;
8     int cuadrado = 0;
9     int cuenta = 0;
10    printf("Ingresa el numero para elevar al cuadrado: \n");
11    scanf("%d", &num);
12    temp = num;
13    while(temp > 0)
14    {
15        if(temp & 1)
16        {
17            cuadrado += num << cuenta;
18        }
19        temp = temp >> 1;
20        cuenta++;
21    }
22    printf("El cuadrado de %d es: %d\n", num, cuadrado);
23    return 0;
24 }
```

Compiler Resources Compile Log Debug Find Results Close

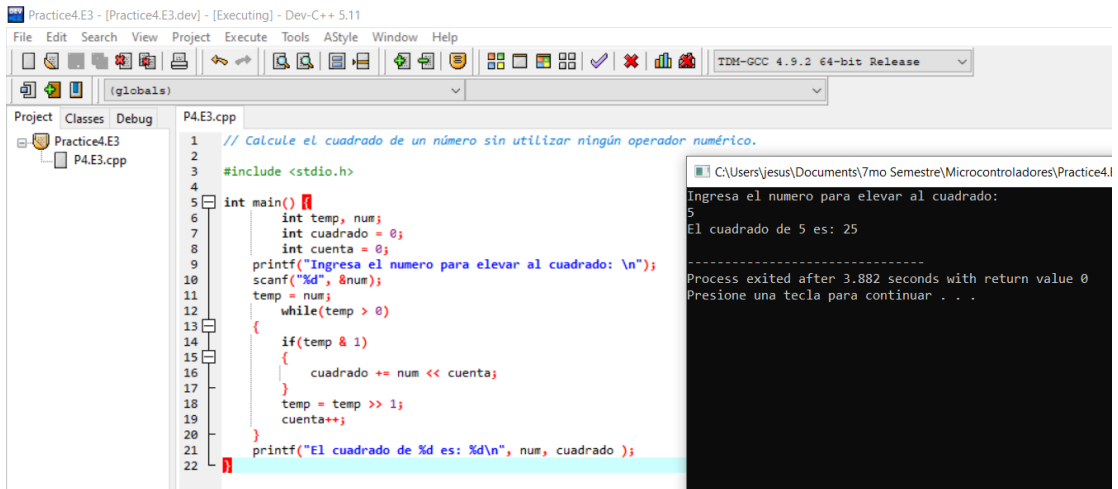
Compilation results...

Errors: 0
Warnings: 0
Output Filename: C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.E3.exe
Output Size: 128.7705078125 KiB
Compilation Time: 0.39s

Line: 22 Col: 2 Sel: 0 Lines: 22 Length: 943 Insert Done parsing in 0.016 seconds

Para lograr elevar al cuadrado utilicé un while que repitiera una operación hasta que se lograra obtener el cuadrado de la variable “num” que es el número introducido por el usuario, para lograr esto hice uso de dos variables, “temp” y “cuenta”. Finalmente, el programa le repite al usuario su número y le muestra su cuadrado.

Funcionamiento:

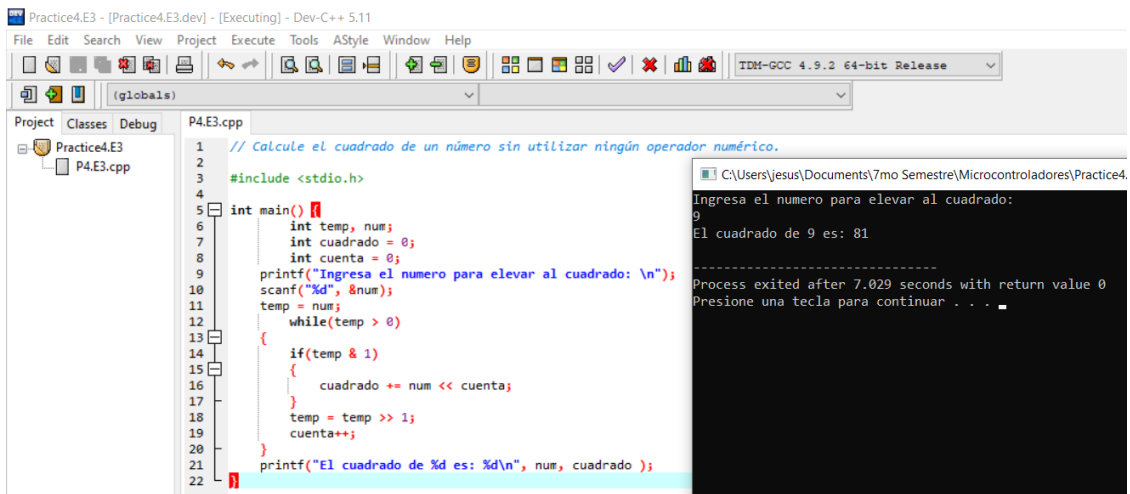


The screenshot shows the Dev-C++ IDE with the file `P4.E3.cpp` open. The code is as follows:

```
1 // Calcule el cuadrado de un número sin utilizar ningún operador numérico.
2
3 #include <stdio.h>
4
5 int main()
6 {
7     int temp, num;
8     int cuadrado = 0;
9     int cuenta = 0;
10    printf("Ingresa el numero para elevar al cuadrado: \n");
11    scanf("%d", &num);
12    temp = num;
13    while(temp > 0)
14    {
15        if(temp & 1)
16        {
17            cuadrado += num << cuenta;
18        }
19        temp = temp >> 1;
20        cuenta++;
21    }
22    printf("El cuadrado de %d es: %d\n", num, cuadrado );
```

The output window shows the following text:

```
C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.1
Ingresa el numero para elevar al cuadrado:
5
El cuadrado de 5 es: 25
-----
Process exited after 3.882 seconds with return value 0
Presione una tecla para continuar . . .
```

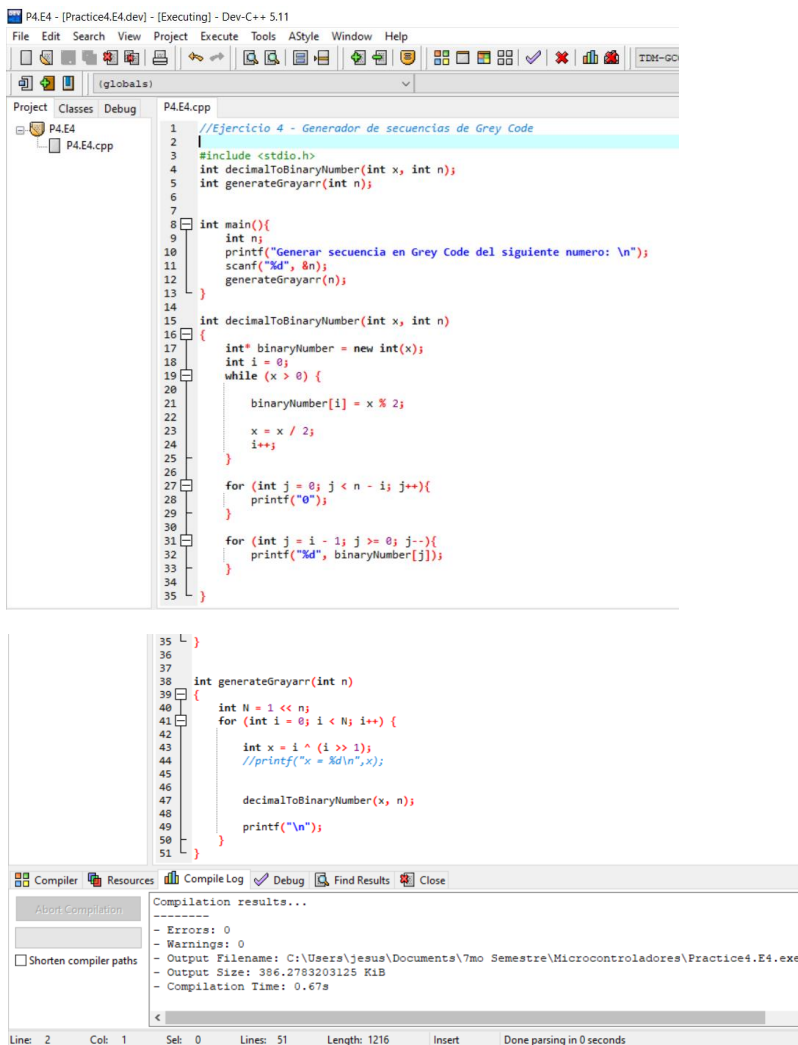


The screenshot shows the Dev-C++ IDE with the file `P4.E3.cpp` open. The code is the same as in the previous screenshot. The output window shows the following text:

```
C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.1
Ingresa el numero para elevar al cuadrado:
9
El cuadrado de 9 es: 81
-----
Process exited after 7.029 seconds with return value 0
Presione una tecla para continuar . . .
```

Ejercicio 4.- Generador de secuencias de Gray Code

Código:



```
1 //Ejercicio 4 - Generador de secuencias de Gray Code
2
3 #include <stdio.h>
4 int decimalToBinaryNumber(int x, int n);
5 int generateGrayarr(int n);
6
7
8 int main(){
9     int n;
10    printf("Generar secuencia en Grey Code del siguiente numero: \n");
11    scanf("%d", &n);
12    generateGrayarr(n);
13 }
14
15 int decimalToBinaryNumber(int x, int n)
16 {
17     int* binaryNumber = new int(x);
18     int i = 0;
19     while (x > 0) {
20
21         binaryNumber[i] = x % 2;
22
23         x = x / 2;
24         i++;
25     }
26
27     for (int j = 0; j < n - i; j++){
28         printf("0");
29     }
30
31     for (int j = i - 1; j >= 0; j--){
32         printf("%d", binaryNumber[j]);
33     }
34 }
35
36
37
38 int generateGrayarr(int n)
39 {
40     int N = 1 << n;
41     for (int i = 0; i < N; i++) {
42
43         int x = i ^ (i >> 1);
44         //printf("x = %d\n", x);
45
46         decimalToBinaryNumber(x, n);
47
48         printf("\n");
49     }
50 }
51 }
```

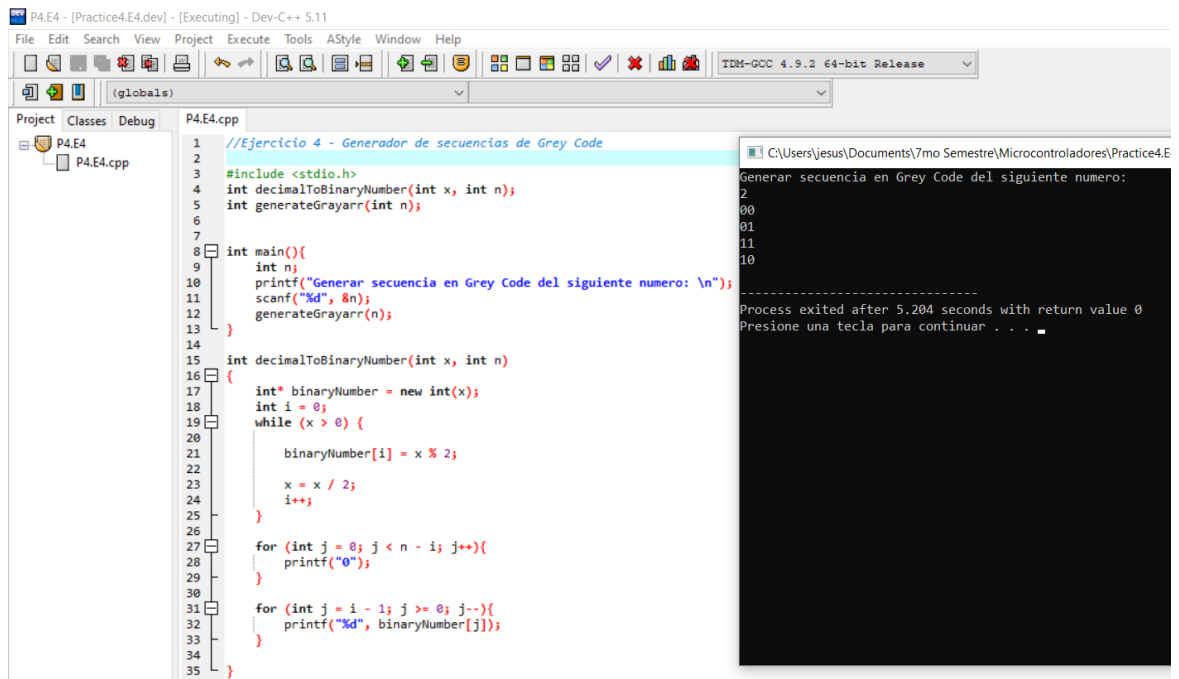
Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\jesus\Documents\7mo Semestre\Microcontroladores\Practice4.E4.exe
- Output Size: 386.2783203125 KiB
- Compilation Time: 0.67s

Line: 2 Col: 1 Sel: 0 Lines: 51 Length: 1216 Insert Done parsing in 0 seconds

Este fue por mucho el código más complicado de hacer, pero con algo de investigación por internet descubrí como hacer que se genera una secuencia en donde 1 bit cambiara en cada patrón sucesivo según el numero decimal que el usuario introdujera.

Funcionamiento:



The screenshot shows a C++ IDE with the file `P4.E4.cpp` open. The code implements a function `decimalToBinaryNumber` and a `main` function that prompts the user for a number and generates a Gray code sequence. The output window shows the sequence for the number 2.

```
//Ejercicio 4 - Generador de secuencias de Grey Code
#include <stdio.h>
int decimalToBinaryNumber(int x, int n);
int generateGrayarr(int n);

int main(){
    int n;
    printf("Generar secuencia en Grey Code del siguiente numero: \n");
    scanf("%d", &n);
    generateGrayarr(n);
}

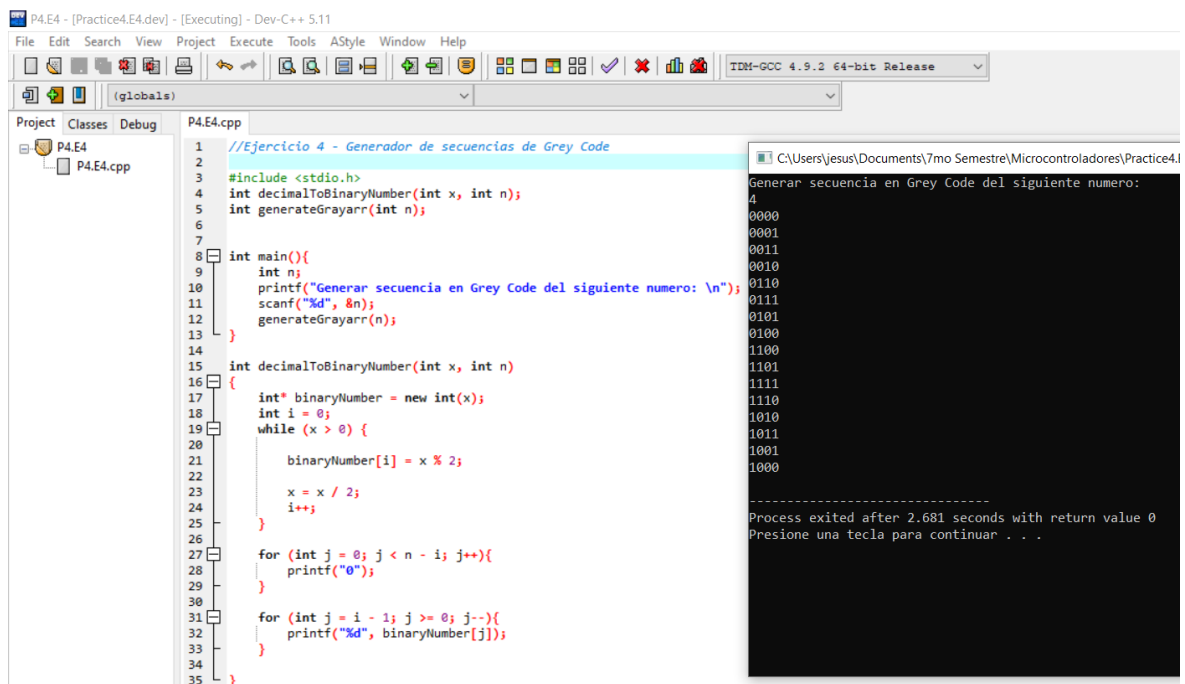
int decimalToBinaryNumber(int x, int n)
{
    int* binaryNumber = new int(x);
    int i = 0;
    while (x > 0) {
        binaryNumber[i] = x % 2;
        x = x / 2;
        i++;
    }

    for (int j = 0; j < n - i; j++){
        printf("0");
    }

    for (int j = i - 1; j >= 0; j--){
        printf("%d", binaryNumber[j]);
    }
}
```

Output:

```
Generar secuencia en Grey Code del siguiente numero:
2
00
01
11
10
-----
Process exited after 5.204 seconds with return value 0
Presione una tecla para continuar . . .
```



The screenshot shows the same C++ IDE with the file `P4.E4.cpp` open. The code is the same as in the first screenshot. The output window shows the sequence for the number 4.

```
//Ejercicio 4 - Generador de secuencias de Grey Code
#include <stdio.h>
int decimalToBinaryNumber(int x, int n);
int generateGrayarr(int n);

int main(){
    int n;
    printf("Generar secuencia en Grey Code del siguiente numero: \n");
    scanf("%d", &n);
    generateGrayarr(n);
}

int decimalToBinaryNumber(int x, int n)
{
    int* binaryNumber = new int(x);
    int i = 0;
    while (x > 0) {
        binaryNumber[i] = x % 2;
        x = x / 2;
        i++;
    }

    for (int j = 0; j < n - i; j++){
        printf("0");
    }

    for (int j = i - 1; j >= 0; j--){
        printf("%d", binaryNumber[j]);
    }
}
```

Output:

```
Generar secuencia en Grey Code del siguiente numero:
4
0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000
-----
Process exited after 2.681 seconds with return value 0
Presione una tecla para continuar . . .
```