**POST LABORATORIO PRACTICA Nro : 6**

|  |  |
| --- | --- |
| **Estudiante 1** | **César Barreto** |
| **Estudiante 2** |  |

**EJERCICIO 1**

1.- Elaborar un programa en Pascal que, usando subprogramas, realice el cálculo de las raices en una ecuación cuadrática. Para ello debe pedir por teclado los valores de a,b, y c que deben ser pasados como parámetros desde el programa principal a un subprograma , realizar en el subprograma los cálculos de las raíces y devolver los resultado obtenidos hacia el programa principal, en donde se deben visualizar. **No se deben solicitar los datos de entrada ni visualizar los resultados en los subprogramas, todos los mensajes de entrada y salida de datos se deben hacer en el programa principal**

El cálculo del discriminante de la ecuación debe realizarse a través de una FUNCTION.

Para calcular las raíces de la ecuación cuadrática se usa la fórmula: ax2 + bx + c = 0 , teniendo en cuenta que :

a) Si “a” es igual a 0 y “b” es igual a 0, imprimiremos un mensaje diciendo que la “Ecuación es degenerada”.

b) Si “a” es igual a 0 y “b” no es igual a 0, existe una raíz única con valor –c/b, en este caso se debe enviar el mensaje “Existe una raíz única” y de seguida el valor de la raíz.

c) En los demás casos, utilizaremos la fórmula: xi = (-b ± √(b2 – 4ac))/(2a)

• La expresión d = b2 – 4ac se denomina discriminante.

• Si “d” es mayor que 0 entonces hay dos raíces reales, que se deben calcular según la fórmula y mostrarse, junto al mensaje “Existen dos raíces reales”.

• Si “d” es igual que 0 entonces hay una raíz única , que se deben calcular “-b/2ª”, en este caso se debe enviar el mensaje “Existe una raíz única”

• Si “d” es menor que 0 entonces hay dos raíces complejas de la forma: J+Qi y J-Qi. Siendo J = –b/2a y Q = √!(b2-4ac)!/(2a), que se se lee como ( la raíz cuadrada del valor absoluto de (b2-4ac))/(2a).

Nota : el carácter √ corresponde a raíz cuadrada

**ANALISIS:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Coeficiente a:  a.  Coeficiente b:  b.  Coeficiente c:  c. | Pedir los datos de entrada.  Verificar el valor de a y b para determinar la formula a utilizar, dependiendo de los siguientes casos:  Si a = 0 y b = 0:  Mostrar salida.  Si a= 0 y b <> 0:  Llamar a la función raízU  Si a<>0 y b <> 0:  Llamar a la función Discriminante  Calcular el valor de las raíces dependiendo del valor del discriminante:  Si el discriminante > 0:  Llamar al procedimiento RaicesReales    Si el discriminante = 0:  Llamar a la función RaizU  Si el discriminante < 0:  Llamar al procedimiento RaicesComplejas.  Mostrar la salida | Mostrar el valor de la o las raíces y un mensaje, según sea el caso. |

**ANALISIS Función discriminante:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Coeficiente a:  ad.  Coeficiente b:  bd.  Coeficiente c:  cd. | Recibir los datos de entrada  Calcular el valor del discriminante.  Enviar el dato de salida al programa principal. | Valor del discriminante:  Discriminante |

**ANALISIS Procedimiento RaicesReales:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Coeficiente a:  aa.  Coeficiente b:  bb.  Discriminante:  d. | Recibir los datos de entrada.  Calcular el valor de las raíces reales utilizando formula de la resolvente cuadrática.  Enviar los datos de salida al programa principal | Valor de las raíces reales:  X1, X2. |

**ANALISIS Función RaizU:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Coeficiente a:  aa.  Coeficiente b:  bb.  Coeficiente c:  cc. | Recibir los datos de entrada.  Calcular el valor de la raíz única según sea el caso:  Si a= 0 y b <> 0.  Si el discriminante = 0:  Calcular el valor de la raíz unica.  Calcular el valor de la raíz unica.  Enviar el dato de salida al programa principal. | Valor de la raíz única dependiendo del tipo de raíz única se va a calcular. |

**ANALISIS Procedimiento RaicesComplejas:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Coeficiente a:  aa.  Coeficiente b:  bb.  Coeficiente c:  cc. | Recibir los datos de entrada.  Calcular el valor de la cantidad real.  Calcular el valor de la cantidad imaginaria.  Enviar los datos de salida al programa principal. | Valor de la raíz compleja:  J.  Valor del numero imaginario:  Q. |

**ALGORITMO EN PSEUDOCÓDIGO:**

**Cabecera:** CalculodeRaices;

**Declaraciones**

  a, b, c, d, RaizR1, RaizR2, RaizCJ, NumIQ: Real;

**//Funcion para el calculo de la raiz unica**

Funcion RaizU (aa, bb, cc: real; Opcion: Entero): Real;

**Inicio de la función**

**//Opcion para determinar que raiz unica es la que se va a calcular**

  En caso de Opcion hacer:

**//cuando b es diferente de 0**

    1:

       RaizU <- -cc/bb;

Fin-1;

**//cuando el discriminante es igual a 0**

    2:

       RaizU <- -bb/(2\*aa);

Fin-2;

  Fin-Caso;

**Fin-Funcion RaizU;**

**//Funcion para el calculo del discriminante**

Funcion Discriminante(ad,bd,cd: Real): Real;

**Inicio de la function**

  Discriminante <- sqr (bd) - 4\*ad\*cd;

**Fin-Funcion Discriminante;**

**//Procedimineto para el calculo de las raices reales**

Procedimiento raicesReales(aa,bb,dd: real; Var x1,x2: real);

**Inicio del procedimiento**

  x1 <- (-bb + sqrt (dd))/(2\*aa);

  x2 <- (-bb - sqrt (dd))/(2\*aa);

**Fin-Procedimiento RaicesReales;**

**//Procedimiento para el calculo de las raices complejas**

Procedimiento RaicesComplejas (aa,bb,cc:Real; Var J,Q:Real);

**Inicio del procedimento**

  J <- -bb/(2\*aa);

  Q <- sqrt (Abs (sqr(bb)-4\*aa\*cc))/(2\*aa);

**Fin-Procedimiento RaicesComplejas;**

**//Programa principal**

**Inicio**

RaizCJ:= 0;

RaizR1:= 0;

RaizR2:= 0;

NumIQ:= 0;

  Escrbir (“Ingrese los valores de A, B y C, segun la formula Ax^2+Bx+C=0:”);

  leer (a, b, c);

**//verficadno los valores de a y b para realizar el calculo de las raices**

  Si (a = 0) Entonces

      Si (b = 0) Entonces

          Escribir (“Ecuacion degenerada.”);

      De lo contrario

          Escribir (“Existe una raiz unica.”);

          Escribir (“Valor de la raiz: “, Llamar RaizU(a,b,c,1));

      Fin-si;

  De lo contrario

    Si (a <> 0) y (b <> 0) entonces

        d <- Llamar Discriminante (a,b,c);

**//Determinando que procedimiento se utilizara para el calculo de las raices**

**//Raices Reales**

        Si (d > 0) Entonces

            Llamar raicesReales (a,b,d,RaizR1,RaizR2);

            Escribir (“Existen dos raices reales.”);

            Escribir (“Valor de la primera raiz (x1): “, RaizR1);

            Escribir (“Valor de la segunda raiz (x2): “, RaizR2);

        Fin si;

**//Raices complejas**

        Si (d < 0) Entonces

            Llamar RaicesComplejas (a,b,c,RaizCJ,NumIQ);

            Escribir ('Existen dos raices complejas.');

 Escribir (“Valor de la primera raiz (x1): “, RaizCJ:0:2, “ + “, NumIQ:0:2, “i”);

Escribir (“Valor de la segunda raiz (x2): “, RaizCJ:0:2, “ – “, NumIQ:0:2, “i”);

        Fin-Si;

**//Raiz unica**

        Si (d = 0) Entonces

              Escribir (“Existe una raiz unica.”);

              writeln (“Valor de la raiz: “, Llamar RaizU(a,b,c,2));

        Fin-Si;

      Fin-Si;

**Fin.**

**PROGRAMA FUENTE:**

Program CalculodeRaices;

Uses Crt;

Var

  a, b, c, d, RaizR1, RaizR2, RaizCJ, NumIQ: Real;

***//Funcion para el calculo de la raiz unica***

*Function* RaizU (aa, bb, cc: real; Opcion: integer): Real;

Begin

***//Opcion para determinar que raiz unica es la que se va a calcular***

  Case opcion Of

***//cuando b es mdiferente de 0***

    1:

       RaizU := -cc/bb;

***//cuando el discriminante es igual a 0***

    2:

       RaizU := -bb/(2\*aa);

  End;

End;

***//Funcion para el calculo del discriminante***

*Function* Discriminante(ad,bd,cd: Real): Real;

Begin

  Discriminante := sqr (bd) - 4\*ad\*cd;

End;

***//Procedimineto para el calculo de las raices reales***

*Procedure* raicesReales(aa,bb,dd: real; Var x1,x2: real);

Begin

  x1 := (-bb + sqrt (dd))/(2\*aa);

  x2 := (-bb - sqrt (dd))/(2\*aa);

End;

***//Procedimiento para el calculo de las raices complejas***

*Procedure* RaicesComplejas (aa,bb,cc:Real; Var J,Q:Real);

Begin

  J := -bb/(2\*aa);

  Q := sqrt (Abs (sqr(bb)-4\*aa\*cc))/(2\*aa);

End;

***//Programa principal***

Begin

RaizCJ:= 0;

RaizR1:= 0;

RaizR2:= 0;

NumIQ:= 0;

  Writeln ('Ingrese los valores de A, B y C, segun la formula Ax^2+Bx+C=0:');

  Readln (a, b, c);

***//verficadno los valores de a y b para realizar el calculo de las raices***

  If (a = 0) Then

    Begin

      If (b = 0) Then

        Begin

          clrscr;

          Writeln ('Ecuacion degenerada.');

        End

      Else

        Begin

          Writeln ('Existe una raiz unica.');

          Writeln ('Valor de la raiz: ', RaizU(a,b,c,1): 0: 2);

        End;

    End

  Else

    If (a <> 0) And (b <> 0) Then

      Begin

        d := Discriminante (a,b,c);

***//Determinando que procedimiento se utilizara para el calculo de las raices***

***//Raices Reales***

        If (d > 0)Then

          Begin

            raicesReales (a,b,d,RaizR1,RaizR2);

            clrscr;

            Writeln ('Existen dos raices reales.');

            Writeln ('Valor de la primera raiz (x1): ', RaizR1:0:2);

            Writeln ('Valor de la segunda raiz (x2): ', RaizR2:0:2);

            readkey;

          End;

***//Raices complejas***

        If (d < 0)Then

          Begin

            RaicesComplejas (a,b,c,RaizCJ,NumIQ);

            clrscr;

            Writeln ('Existen dos raices complejas.');

            Writeln ('Valor de la primera raiz (x1): ', RaizCJ:0:2, ' + ', NumIQ:0:2, 'i');

            writeln ('Valor de la segunda raiz (x2): ', RaizCJ:0:2, ' - ', NumIQ:0:2, 'i');

            Readkey;

          End;

***//Raiz unica***

        If (d = 0) Then

          Begin

            clrscr;

            Writeln ('Existe una raiz unica.');

            writeln ('Valor de la raiz: ', RaizU(a,b,c,2): 0: 2);

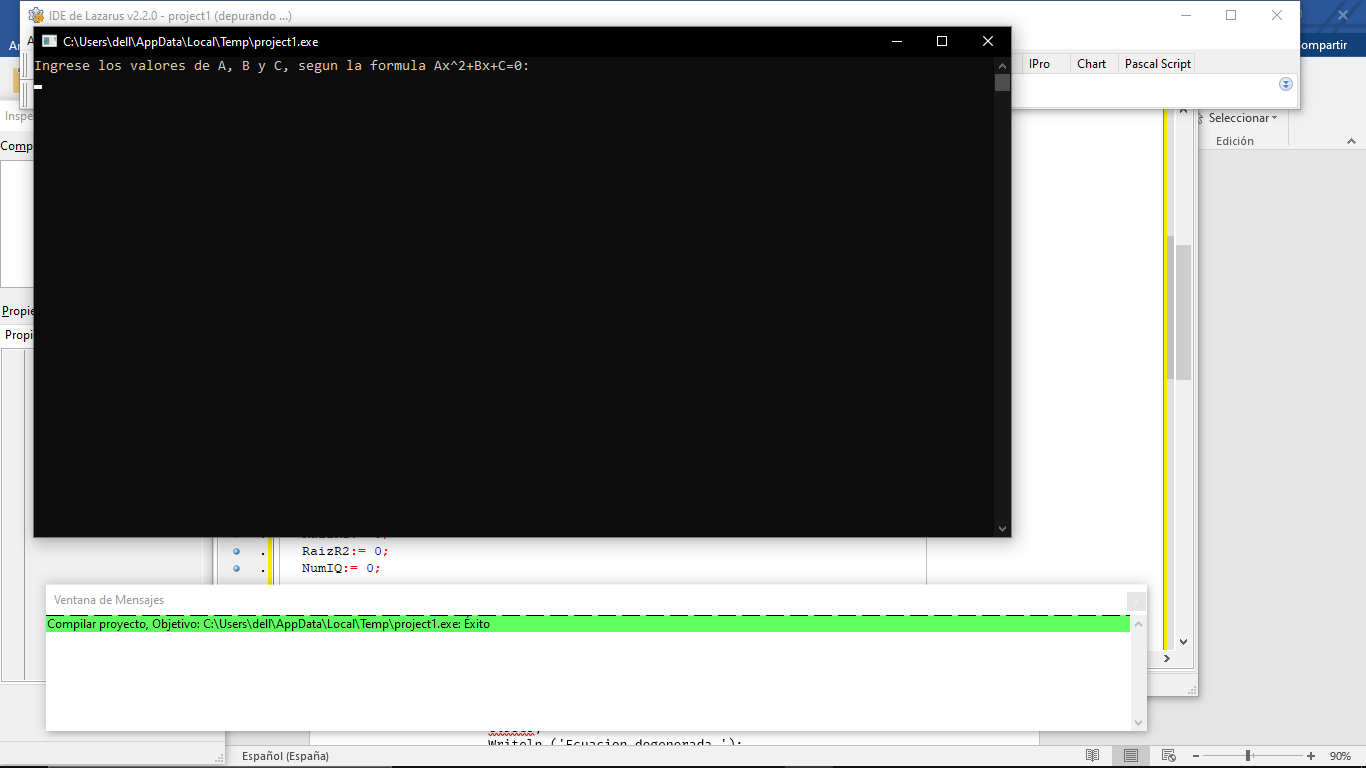
            Readkey;

          End;

      End;

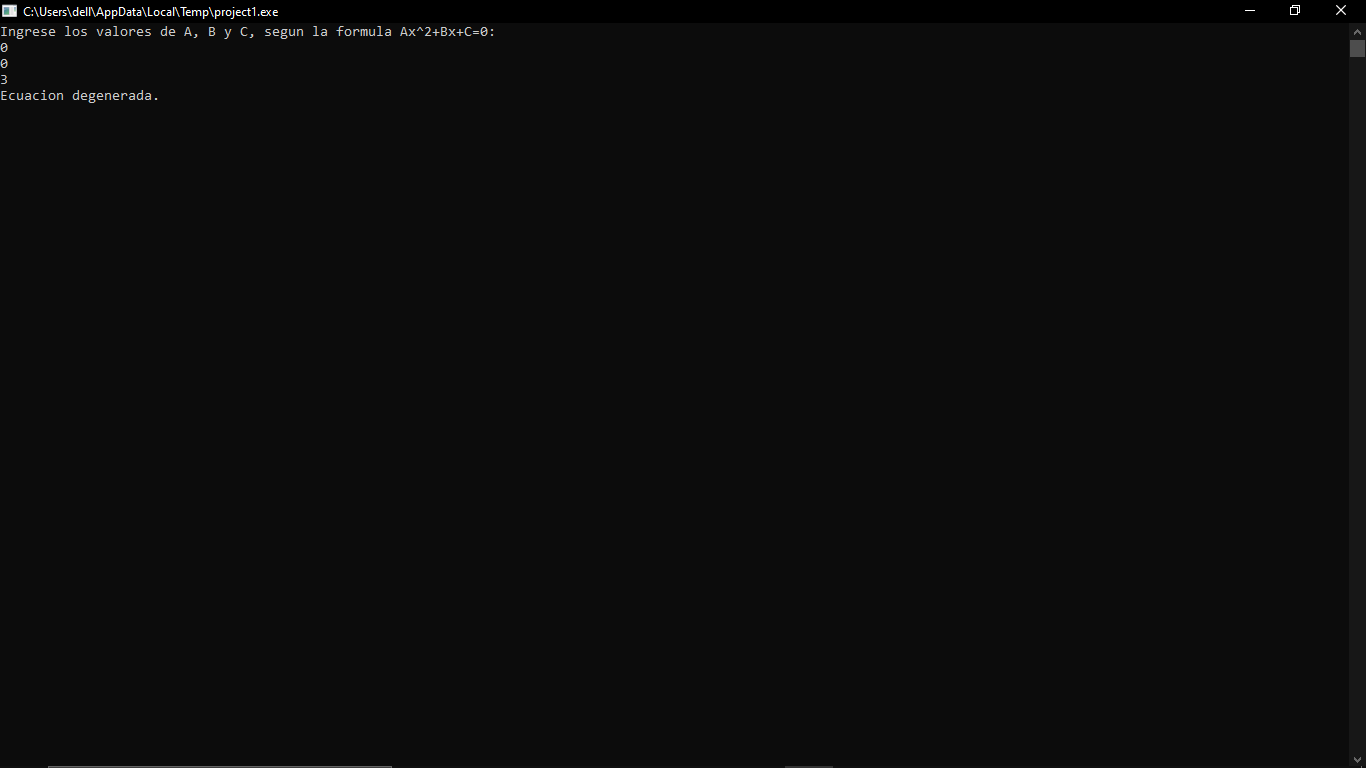
End.

**CAPTURA DE PANTALLA CON RESULTADO DE COMPILACIÓN:**

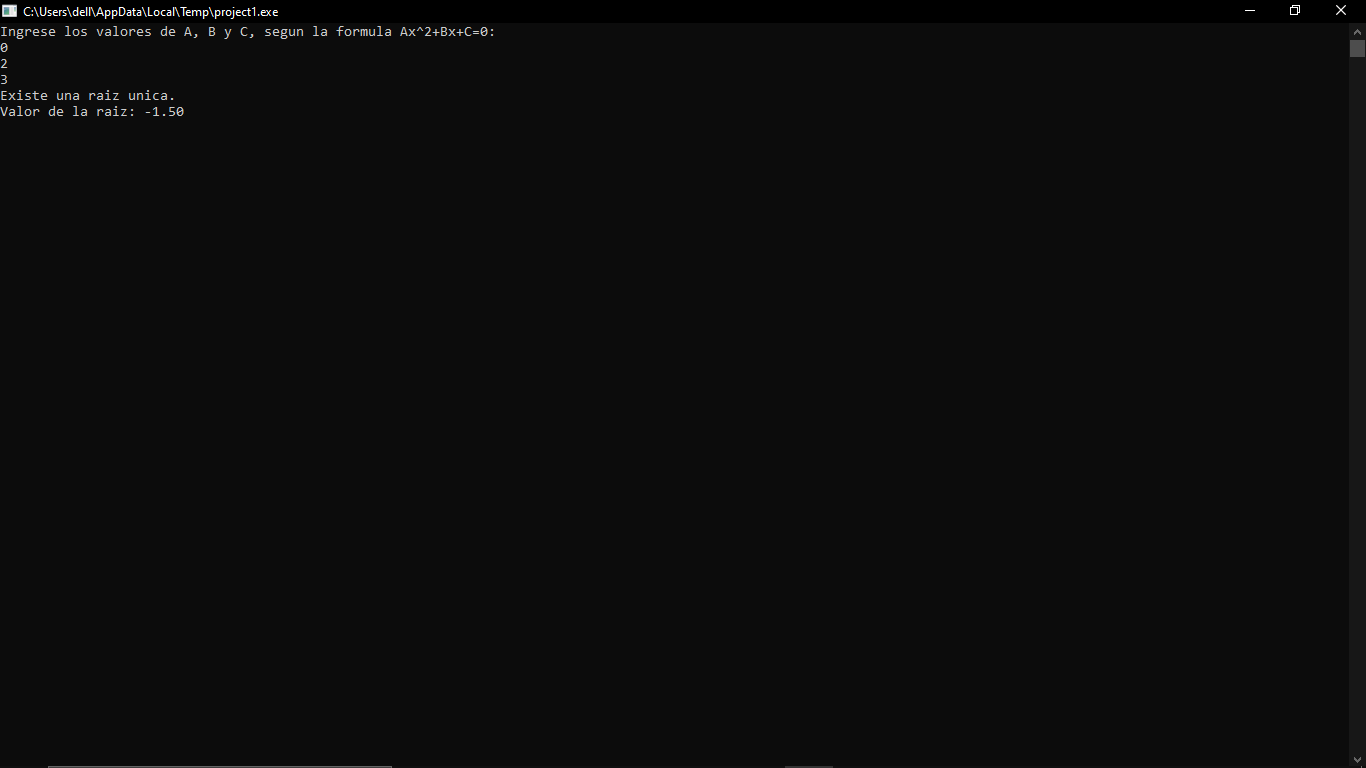


**CAPTURA DE PANTALLAS CON RESULTADO DE EJECUCIÓN (CORRIDAS)**

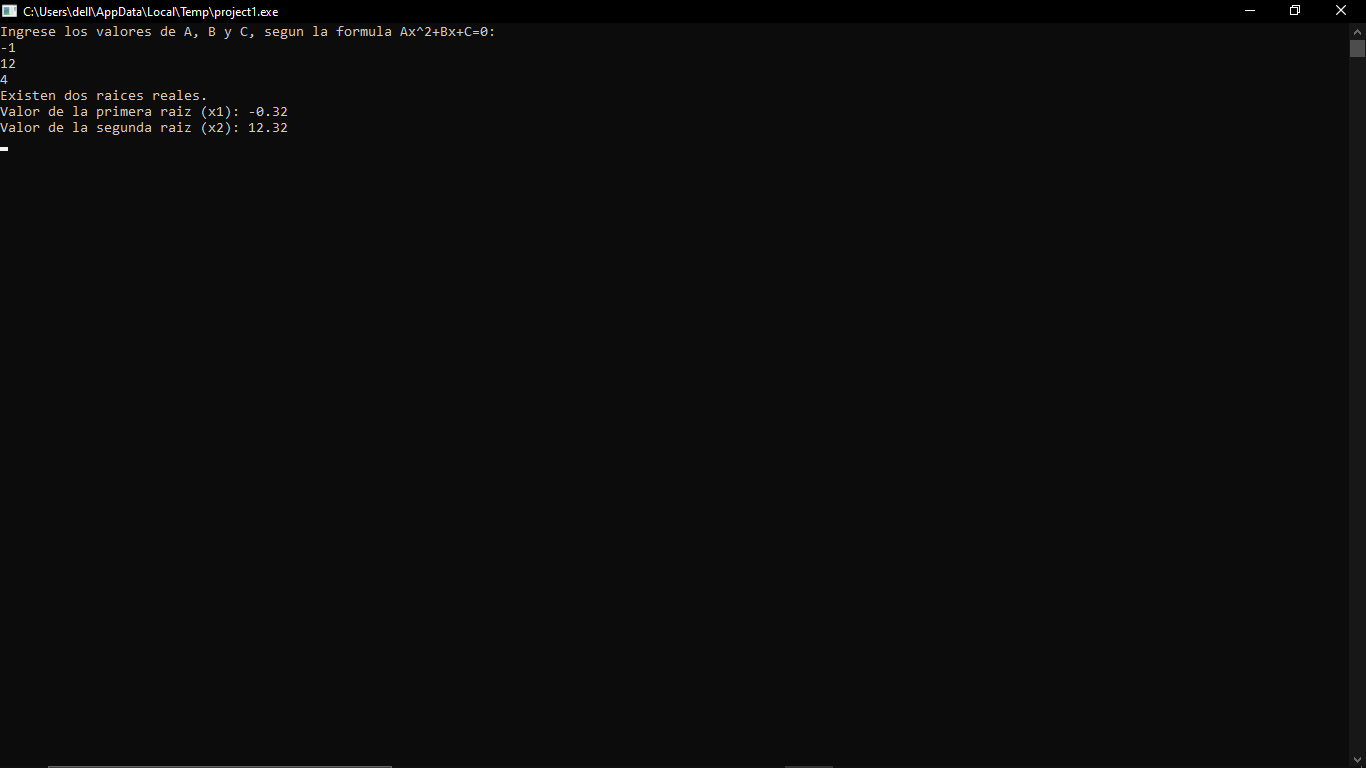
**Ejemplo 1:**



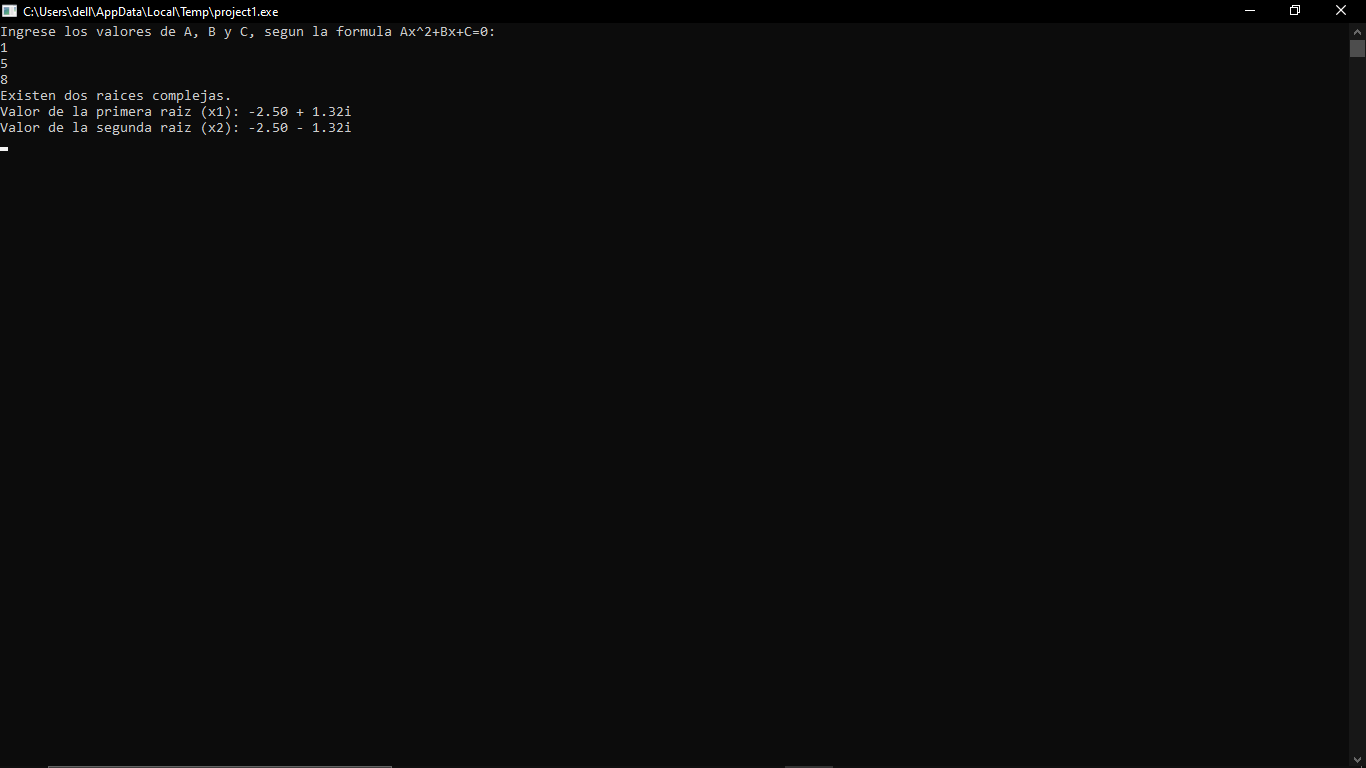
**Ejemplo 2:**



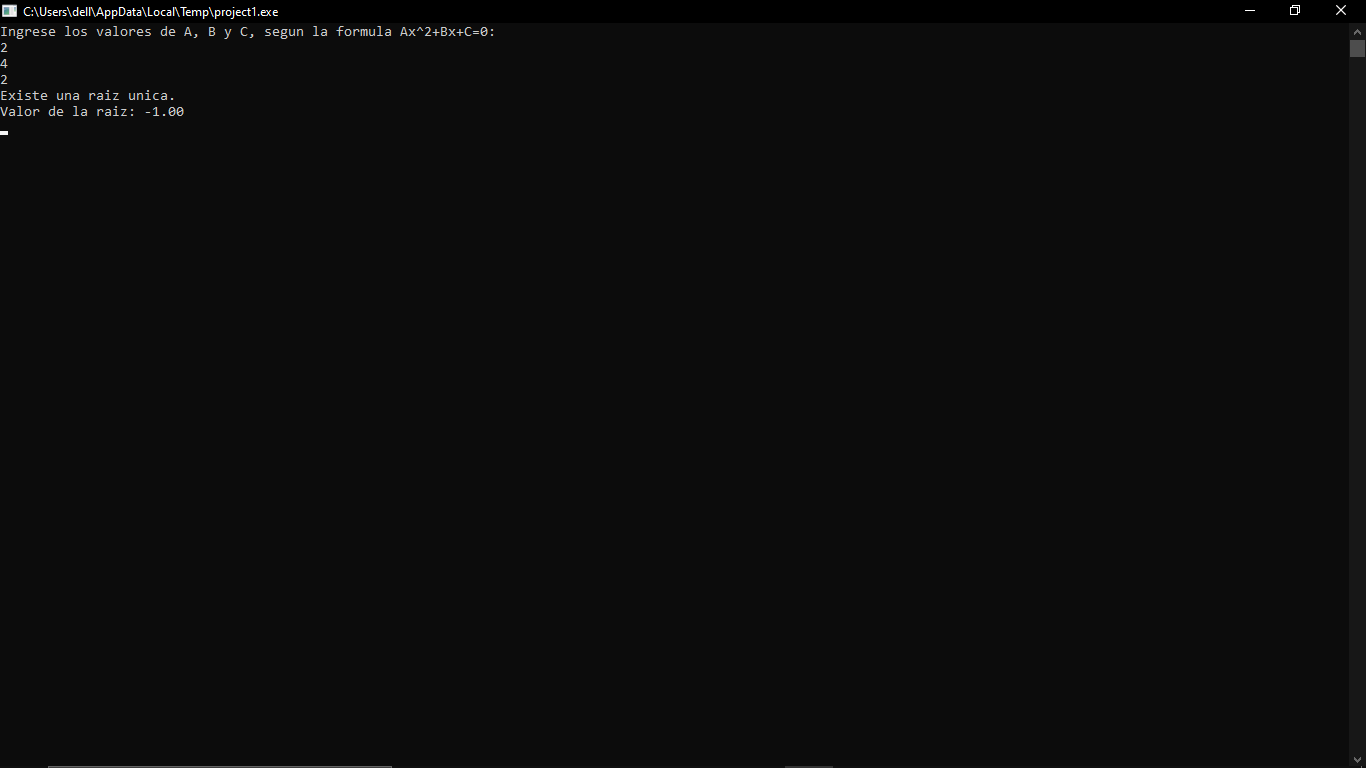
**Ejemplo 3:**



**Ejemplo 4:**



**Ejemplo 5:**



**EJERCICIO 2**

2.- Elabore un programa en Pascal, UTILIZANDO SUBPROGRAMAS **(por lo menos un procedimiento y una función)**, para determinar si un número dado por pantalla es automórfico. Un número automórfico es aquel que al elevarlo sistemáticamente a una potencia mayor que cero, las últimas cifras del resultado corresponden al mismo número. Hacer el programa, para probar que un número es automórfico con un máximo de una potencia N dada. Ejemplo: 76 es automorfico para una potencia N= 4 porque: 76^1= 76; 76^2 = 5776; 76^3 = 438976; 76^4 = 33362176. Fíjese que al elevar el 76 a las 4 potencias sucesivas su resultado teminan en 76. En este caso debe mostrar el resultado y si es o no automórfico.

**Nota :Los números deben leerse en el programa principal y pasarlo al subprograma quien debe hacer los cálculos y devolver los resultados, para que el programa principal los muestre.**

**ANALISIS:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Numero base:  Numero.  La potencia:  Potencia. | Pedir los datos de entrada.  Llamar al procedimiento Descomposicion:  Determinar si es o no es automorfico  Mostrar los datos de salida. | Mostrar el número elevado.  Mostrar si es o no automorfico. |

**ANALISIS Procedimiento Descomposicion:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Numero Base:  Num.  Potencia:  Pot. | Recibir los datos de entrada  Llamar a la función elevar.  Determinar la cantidad de dígitos que posee el numero base.  Elevar un divisor de base 10 a la potencia de la cantidad de los dígitos del número.  Llamar a la función resto.  Enviar los datos de salida al programa principal. | Enviar el valor del resto del número elevado. |

**ANALISIS Funcion Elevar:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Numero base:  N.  Potencia:  P. | Recibir los datos de entrada  Elevar el número base a la potencia dada.  Enviar el dato de salida al procedimiento Descomposición. | el valor del número elevado a la potencia dada. |

**ANALISIS Funcion Resto:**

|  |  |  |
| --- | --- | --- |
| **Entradas** | **Procesos** | **Salida** |
| Numero elevado:  NE.  Divisor:  Divi | Recibir los datos de entrada.  Calcular el resto del numero elevado.  Enviar el dato de salida al procedimiento Descomposicion. | El valor del resto obtenido. |

**ALGORITMO EN PSEUDOCÓDIGO:**

**Cabecera** NumeroAutomorfico;

**Declaraciones**

  Numero, Potencia, resto: Entero;

***//Funcion para elevar un numero dado a una potencia dada***

*Funcion* Elevar (N,P: Entero): Entero largo;

**Inicio de la funcion**

  elevar <- N\*\*P;

**Fin-Funcion Elevar;**

**//Funcion para obtener el resto del numero elevado**

*Funcion* Resto\_NE (NE, Divi: entero): Entero;

**Inicio de la funcion**

  Resto\_NE<- NE Mod Divi;

**Fin-Funcion Resto\_NE;**

**//Procedimiento para descomponer y llamar a las 2 funciones anteriores**

*Procedimiento* Descomposicion (Num, Pot: Entero; Var Rest: Entero);

**Declaraciones del procedimiento**

  Num\_Elev, divisor, Cant\_Dig: Entero;

**Inicio del procedimiento**

  Cant\_Dig <- 1;

  Num\_Elev <- Llamar Elevar (Num, Pot);

  Repetir Mientras (Num Div 10 <> 0) Hacer

      Cant\_Dig <- Cant\_Dig + 1;

      Num <- Num Div 10;

  Fin-RM;

  divisor <- 10\*\*Cant\_Dig;

  Rest <- Llamar Resto\_NE (Num\_Elev, Divisor);

**Fin-Procedimiento Descomposicion;**

**//Programa principal**

**Inicio**

Resto<- 0;

  Escribir (“Ingrese el numero entero a elevar:”);

  Leer (Numero);

  Escribir (“Ingrese la potencia (Mayor que cero) a la cual se elevara el numero:”);

  Leer (Potencia);

  Llamar Descomposicion (Numero, Potencia, Resto);

**//Determinando si es automorfico o no**

  Si (Resto = numero) Entonces

      Escribir (Numero,” elevado a la “, Potencia, “ es: “, Llamar Elevar (Numero,Potencia));

      Escribir (“Su numero es automorfico”);

  De lo contrario

      Escribir (Numero,” elevado a la “, Potencia, “ es: “, Llamar Elevar (Numero,Potencia));

      escrbir (“Su numero no es automorfico”);

  Fin-Si;

**Fin.**

**PROGRAMA FUENTE:**

Program NumeroAutomorfico;

Uses Crt, math;

Var

  Numero, Potencia, resto: Integer;

***//Funcion para elevar un numero dado a una potencia dada***

*Function* Elevar (N,P: integer): longint;

Begin

  elevar := N\*\*P;

End;

***//Funcion para obtener el resto***

*Function* Resto\_NE(NE, Divi: integer): integer;

Begin

  Resto\_NE := NE Mod Divi;

End;

***//Procedimiento para descomponer el numero base y llamar a las 2 funciones anteriores***

*Procedure* Descomposicion(Num, Pot: integer; Var Rest: integer);

Var

  Num\_Elev, divisor, Cant\_Dig: Integer;

Begin

  Cant\_Dig := 1;

  Num\_Elev := Elevar (Num, Pot);

  While (Num Div 10 <> 0) Do

    Begin

      Cant\_Dig := Cant\_Dig + 1;

      Num := Num Div 10;

    End;

  divisor := 10\*\*Cant\_Dig;

  Rest := Resto\_NE (Num\_Elev, divisor);

End;

***//Programa principal***

Begin

Resto:= 0;

  Writeln ('Ingrese el numero entero a elevar:');

  Readln (Numero);

  Writeln ('Ingrese la potencia (Mayor que cero) a la cual se elevara el numero:');

  Readln (Potencia);

  Descomposicion (Numero, Potencia, Resto);

***//Determinando si es automorfico o no***

  If (Resto = numero) Then

    Begin

      Writeln (Numero,' elevado a la ', Potencia, ' es: ', Elevar (Numero,Potencia));

      Writeln ('Su numero es automorfico');

      Readkey;

    End

  Else

    Begin

      Writeln (Numero,' elevado a la ', Potencia, ' es: ', Elevar (Numero,Potencia));

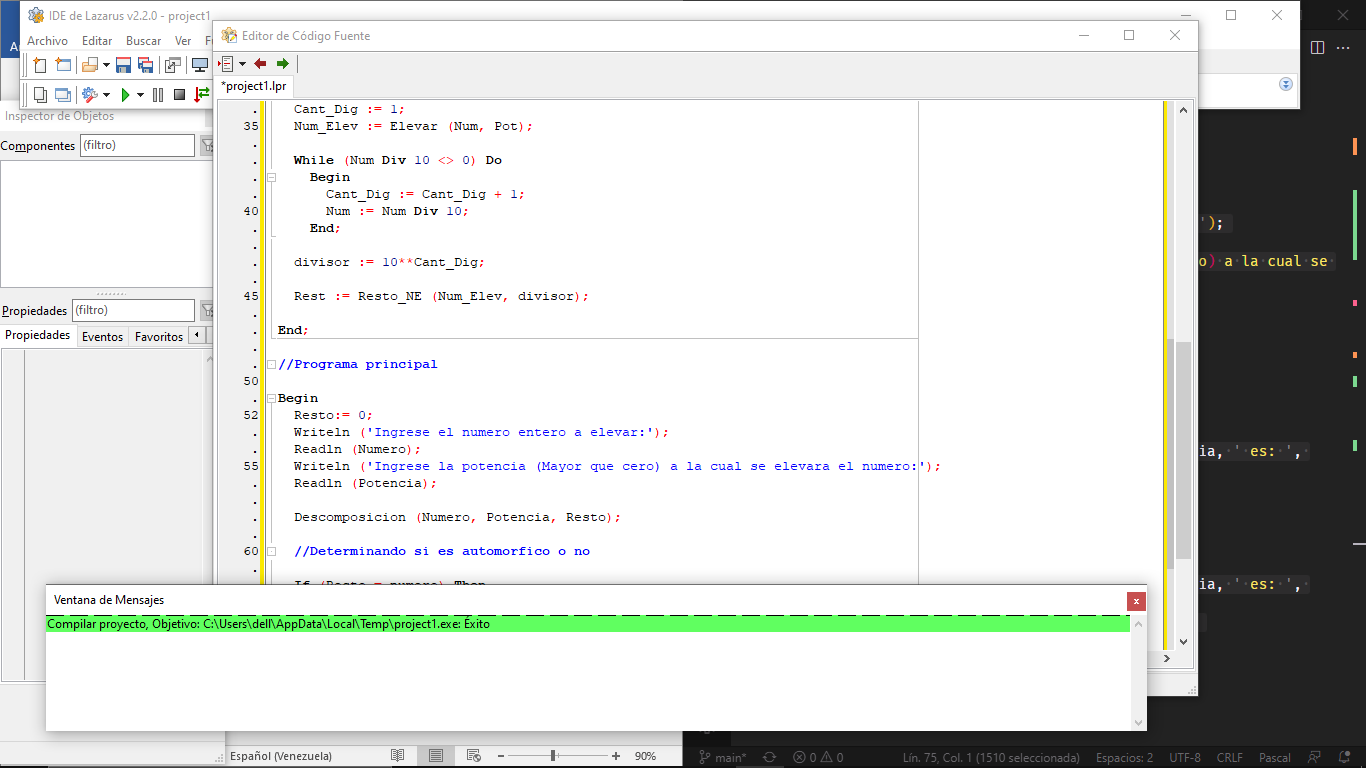
      Writeln ('Su numero no es automorfico');

      Readkey;

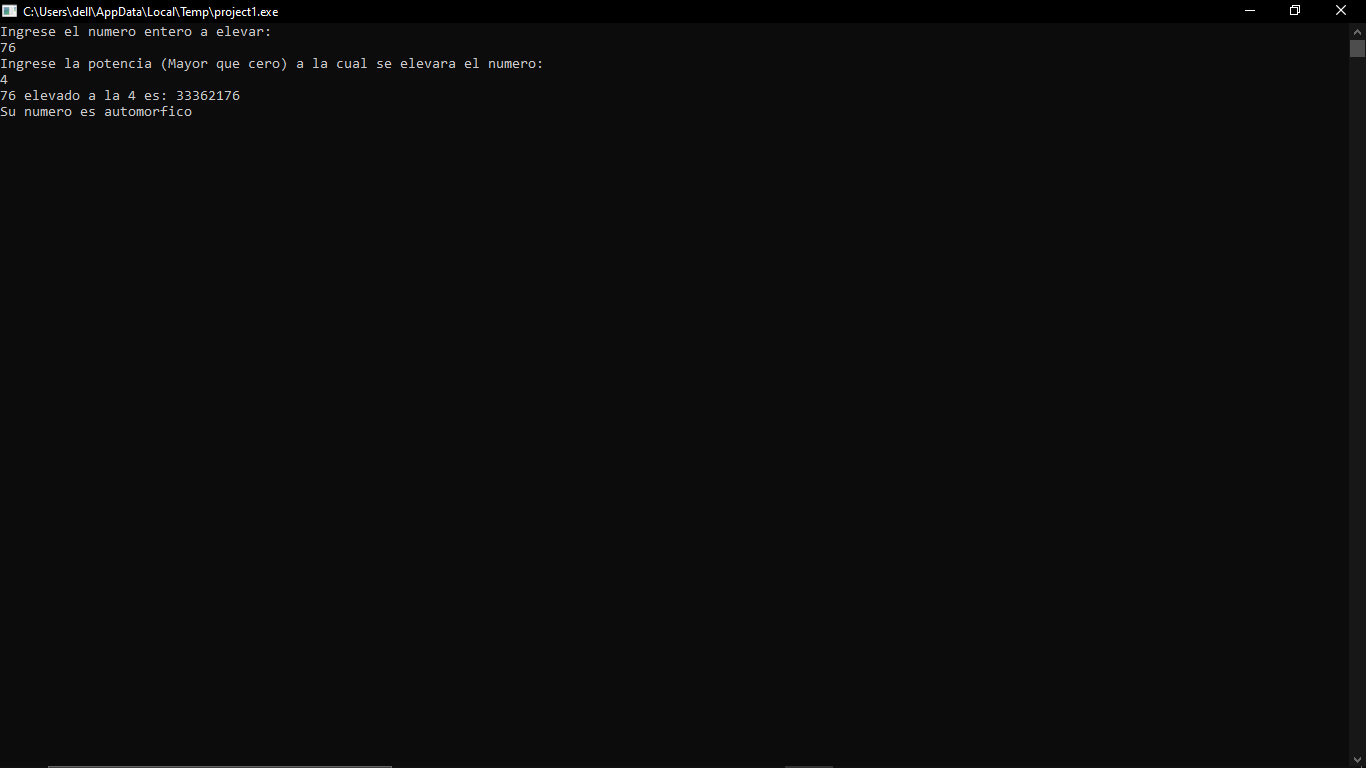
    End;

End.

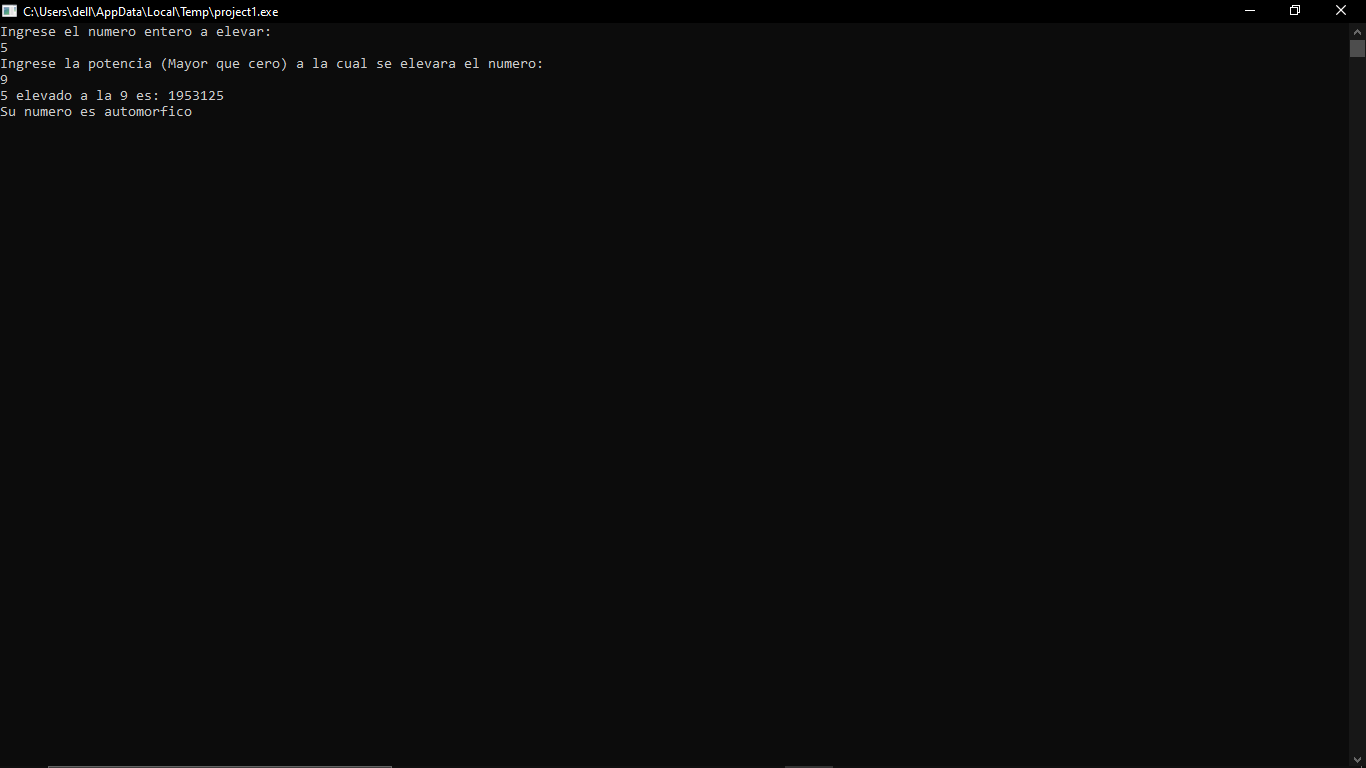
**CAPTURA DE PANTALLA CON RESULTADO DE COMPILACIÓN:**



**CAPTURA DE PANTALLAS CON RESULTADO DE EJECUCIÓN (CORRIDAS) Ejemplo 1:**



**Ejemplo 2:**



**Ejemplo 3:**

