INTRODUCCIÓN AL BUCLE “DO”: SERIES NUMÉRICAS

* 1 + 2 + 3 + 4 + 5 + 6 … + n
* 1 + 3 + 5 + 7 + 9 … + (2 \* n - 1)
* 1 \* 2 \* 3 \* 4 \* 5 \* 6 … \* n

**“sumatorio”**

**“sumatorio\_impares”**

**“factorial”**

Yago Pego Martínez ([yago.pego.martinez@alumnos.upm.es](mailto:yago.pego.martinez@alumnos.upm.es)) Evaristo de Vega Galindo ([evaristo.devega.galindo@alumnos.upm.es](mailto:evaristo.devega.galindo@alumnos.upm.es))

**ESPECIFICACIONES**

**Enunciado**: escribir, compilar y ejecutar tres programas distintos que permitan calcular la suma de los primeros “n” números naturales, los primeros “n” números naturales impares y el factorial de un número. Se deberán realizar capturas de pantalla señalando cada parte en la construcción de los programas.

**Objetivo**: introducir el bucle “do” y considerar las limitaciones numéricas de la variable “integer”.

**INTRODUCCIÓN TEÓRICA**

En el primer programa nos encontramos ante una serie divergente de la forma: an = n + (n-1) + (n-2) … + 2 + 1. Aunque en lenguaje de programación es fácil expresar esta operación a partir del bucle “do”, la fórmula es útil para todo “n” y es típicamente utilizada, de manera más sencilla. Además, aunque esta aplicación trascienda nuestro conocimiento, se ha demostrado que la suma de todos los números naturales es igual a .

El segundo caso nos propone realizar un programa que sume los “n” primeros números naturales impares. Esta serie, como la anterior, tiene también una equivalencia simplificada, equivaliendo la suma de los “n” primeros números naturales al cuadrado de “n”.

“factorial” es el tercer programa y, como su nombre indica, nos permite conocer el valor del producto de un número natural “n” y sus anteriores. Es comúnmente representado con el signo de exclamación (4! = 4\*3\*2\*1 = 24). A pesar de que es algo redundante también se puede calcular el factorial de un número como el producto de ese número y el factorial del anterior.

**BIBLIOTECA DE VARIABLES**

En los tres programas “sumatorio”, “sumatorio\_impares” y “factorial” se usan las mismas variables, todas de tipo entera. Estos datos pueden representar un subconjunto finito de números enteros y el número máximo depende del tamaño del espacio usado.

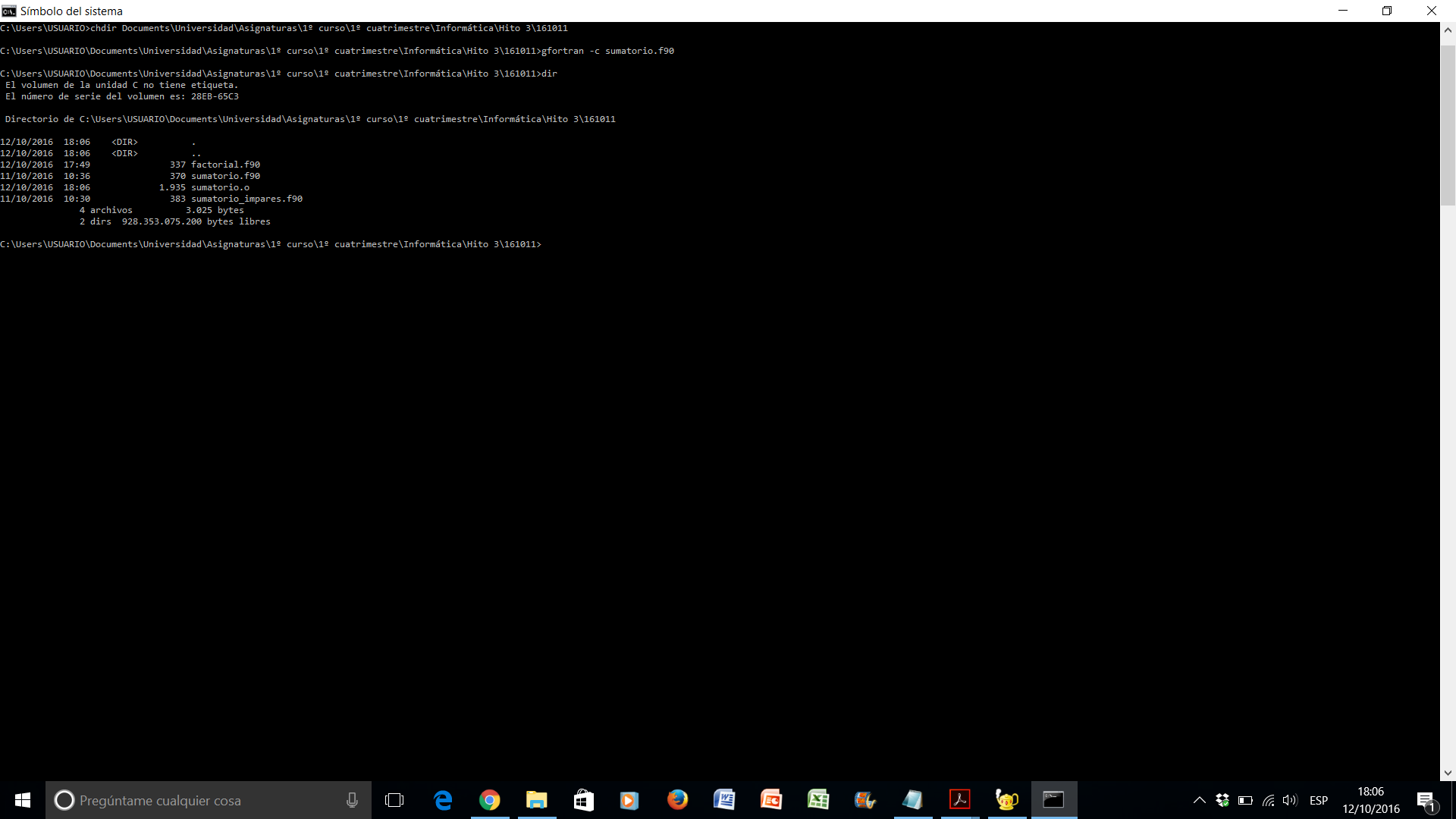
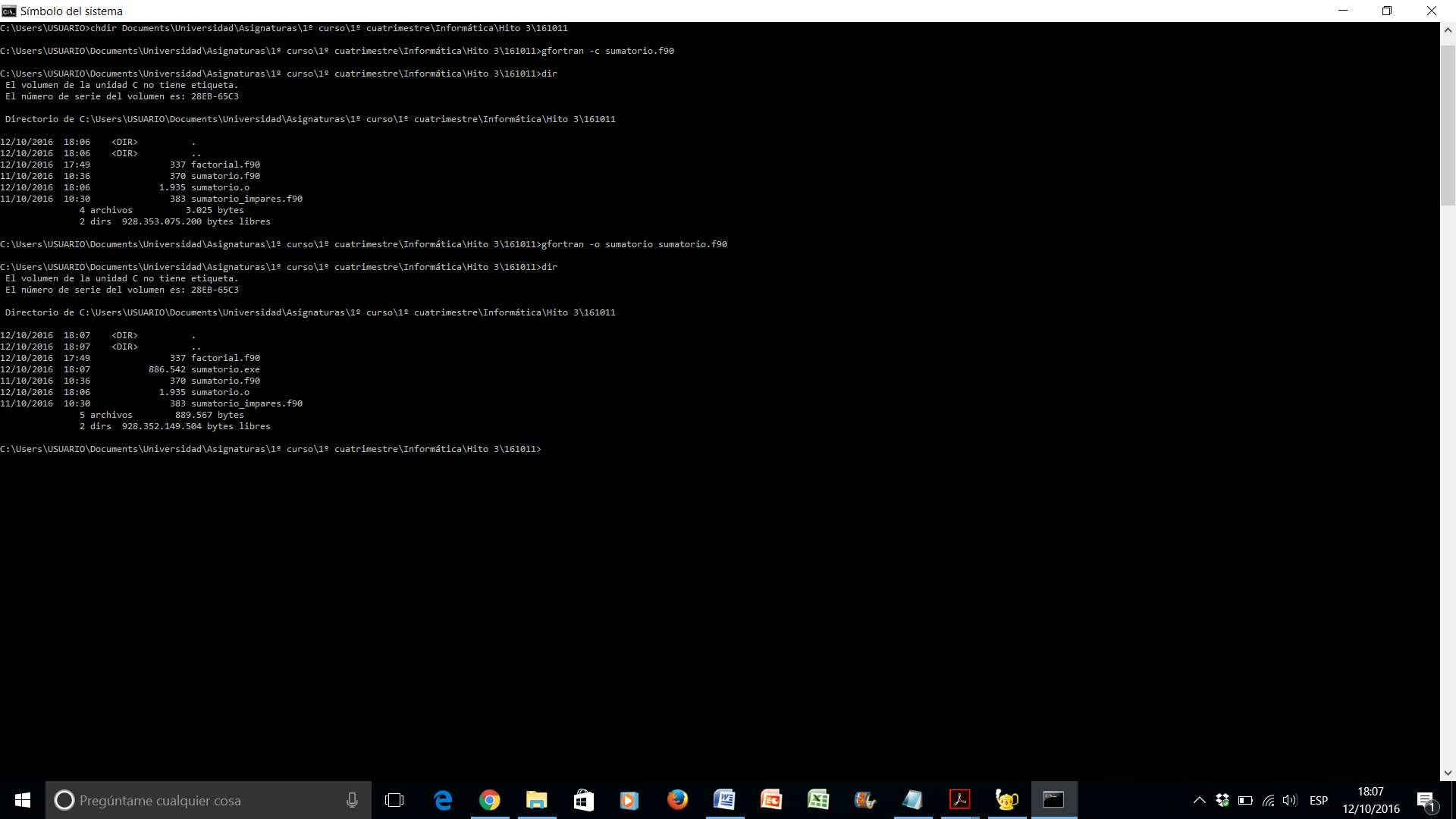
La primera (n) se refiere, como si de una sucesión se tratara, al término en enésima posición.

A la segunda, que hemos denominado “suma” o “factorial” según el programa, simplemente se le aplicará el valor de la sentencia (suma1 = n + (n-1) + (n-2) … + 2 +1; factorial = n!). Esta variable puede llegar a presentar problemas si el número “n” introducido es muy alto. Siendo el identificador de clase predeterminado 4, el máximo valor para “n” en el programa “factorial” será 12, pues 13! sobrepasaría los 2,15\*109, que es el límite.

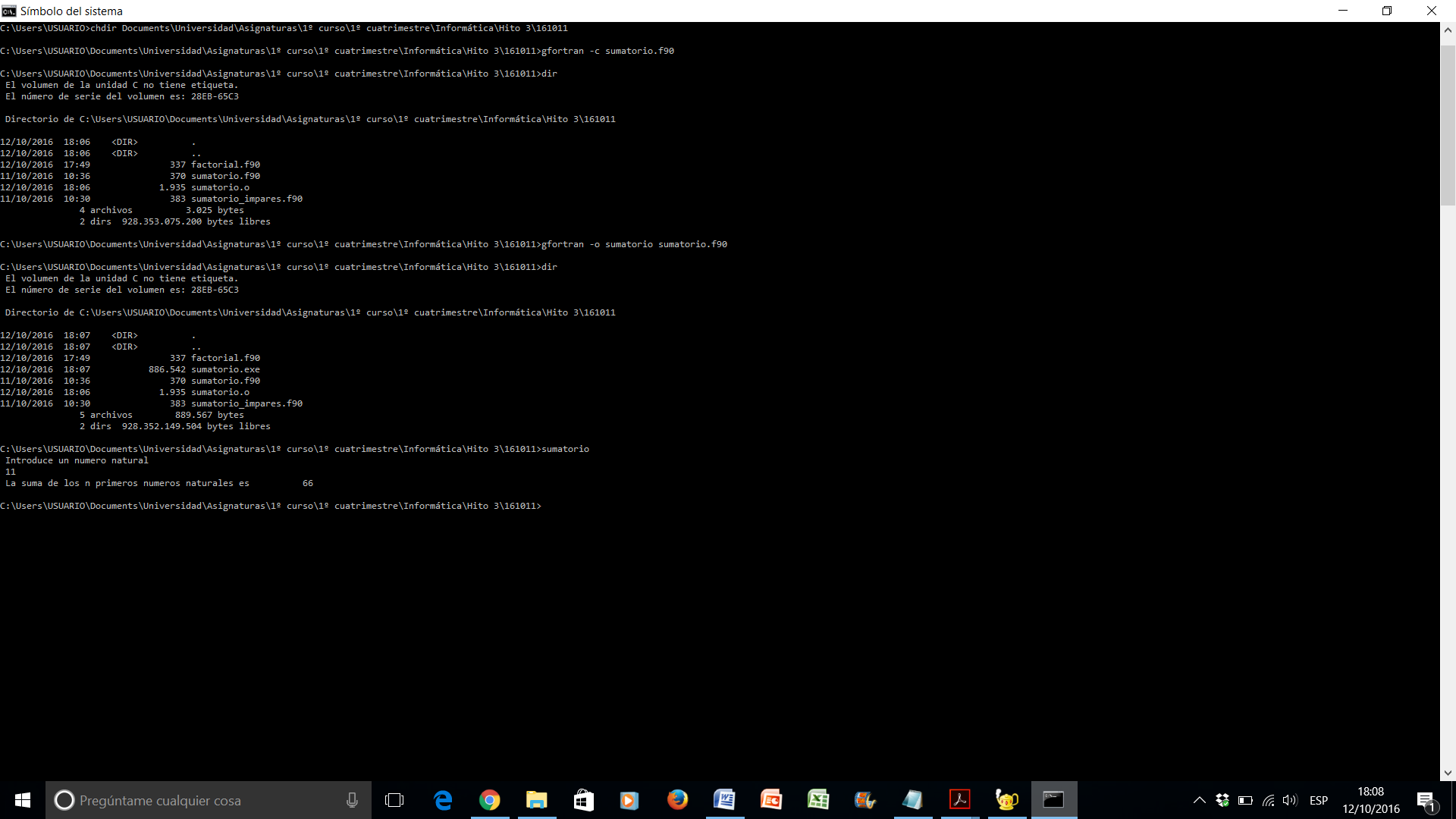
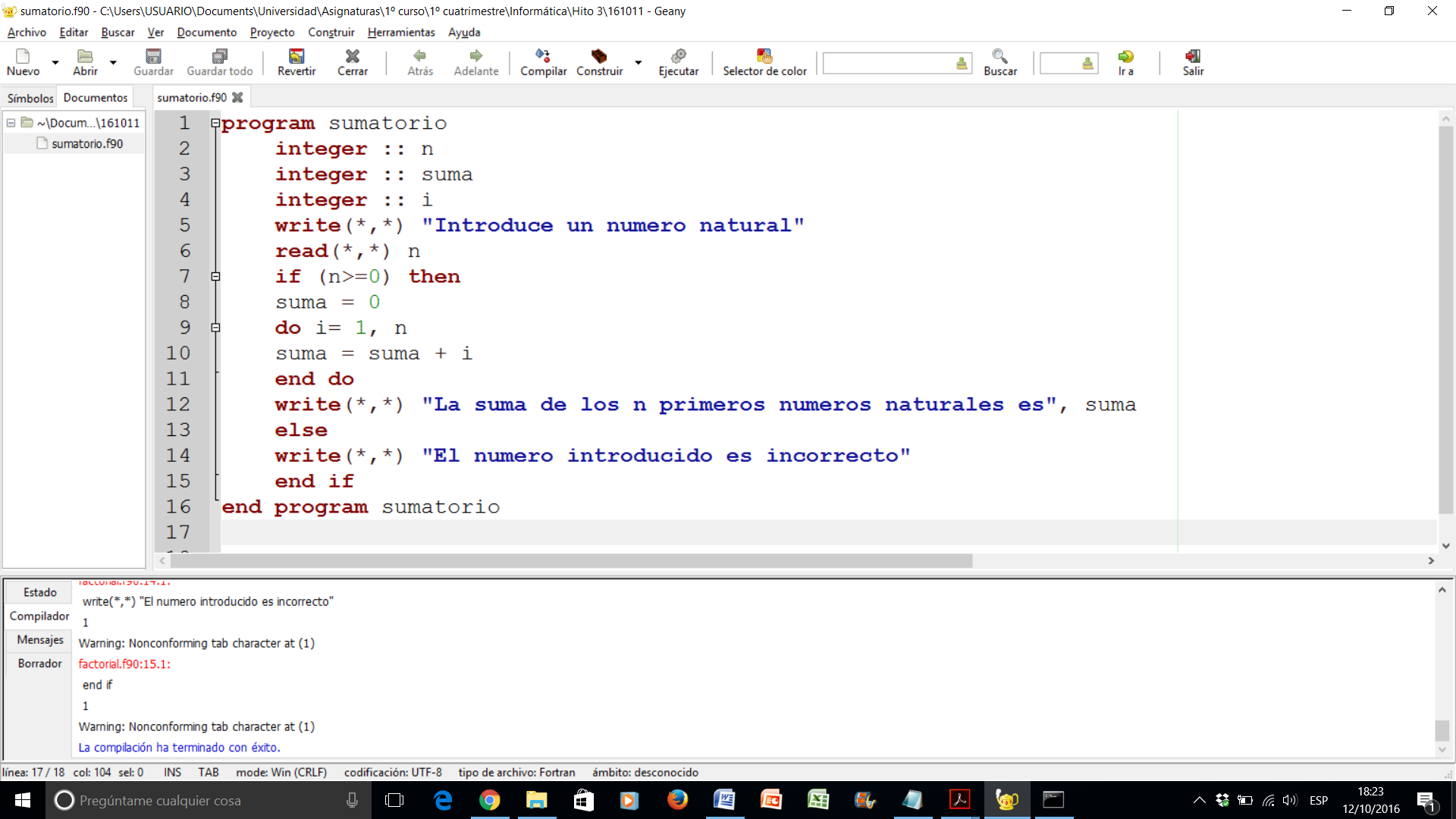
La tercera variable entera es “i”. Esta describe cuál es la operación a realizar, utilizando el valor de la “n” (véase el uso de “i” en las escrituras de los programas en *Geany*).

**RESULTADOS**

*PROGRAMA 1: “sumatorio”*

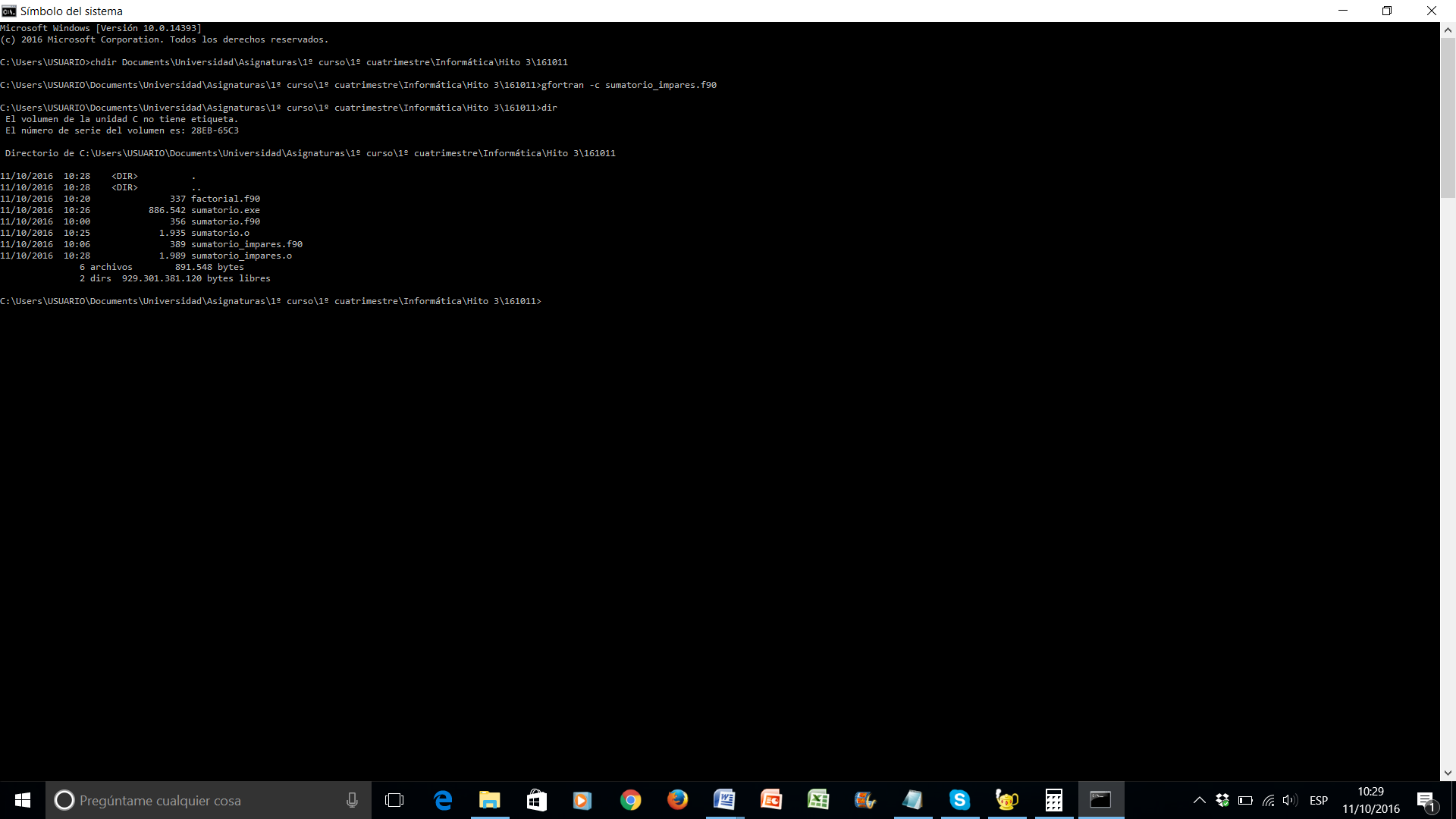


Ejecución del programa “sumatorio” en ‘Símbolo del sistema’



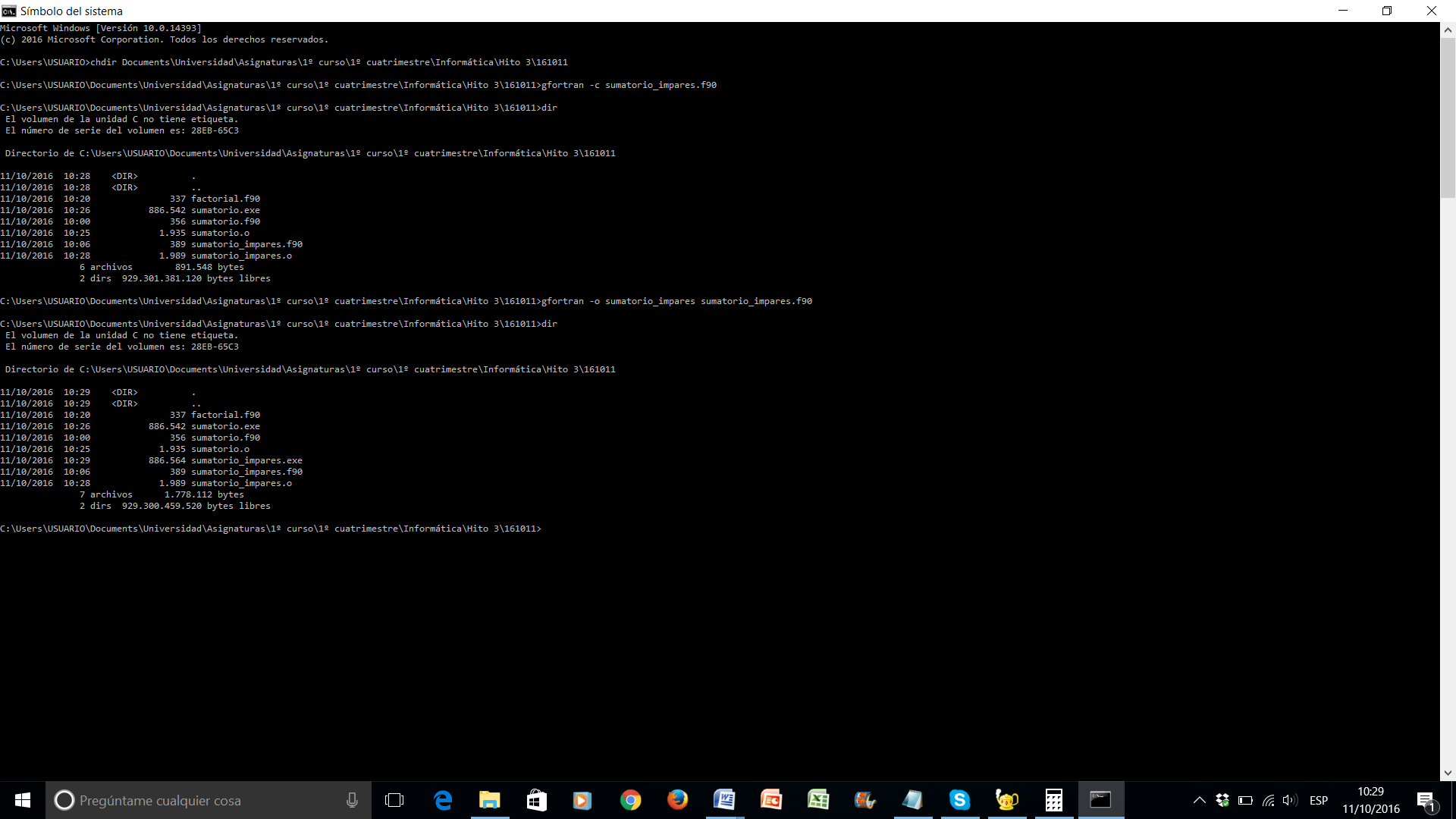
Escritura del programa “sumatorio” en el editor *Geany*

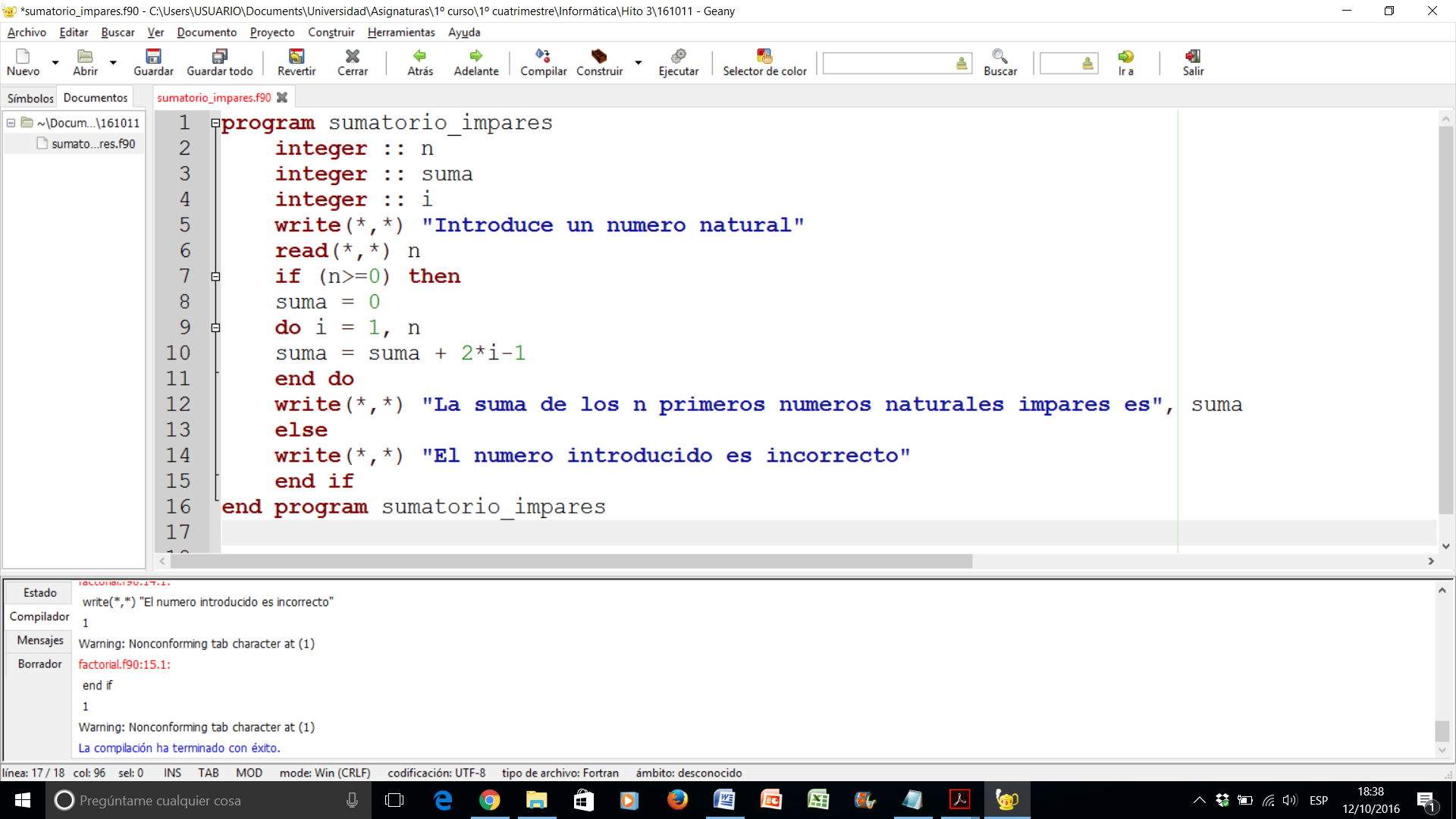
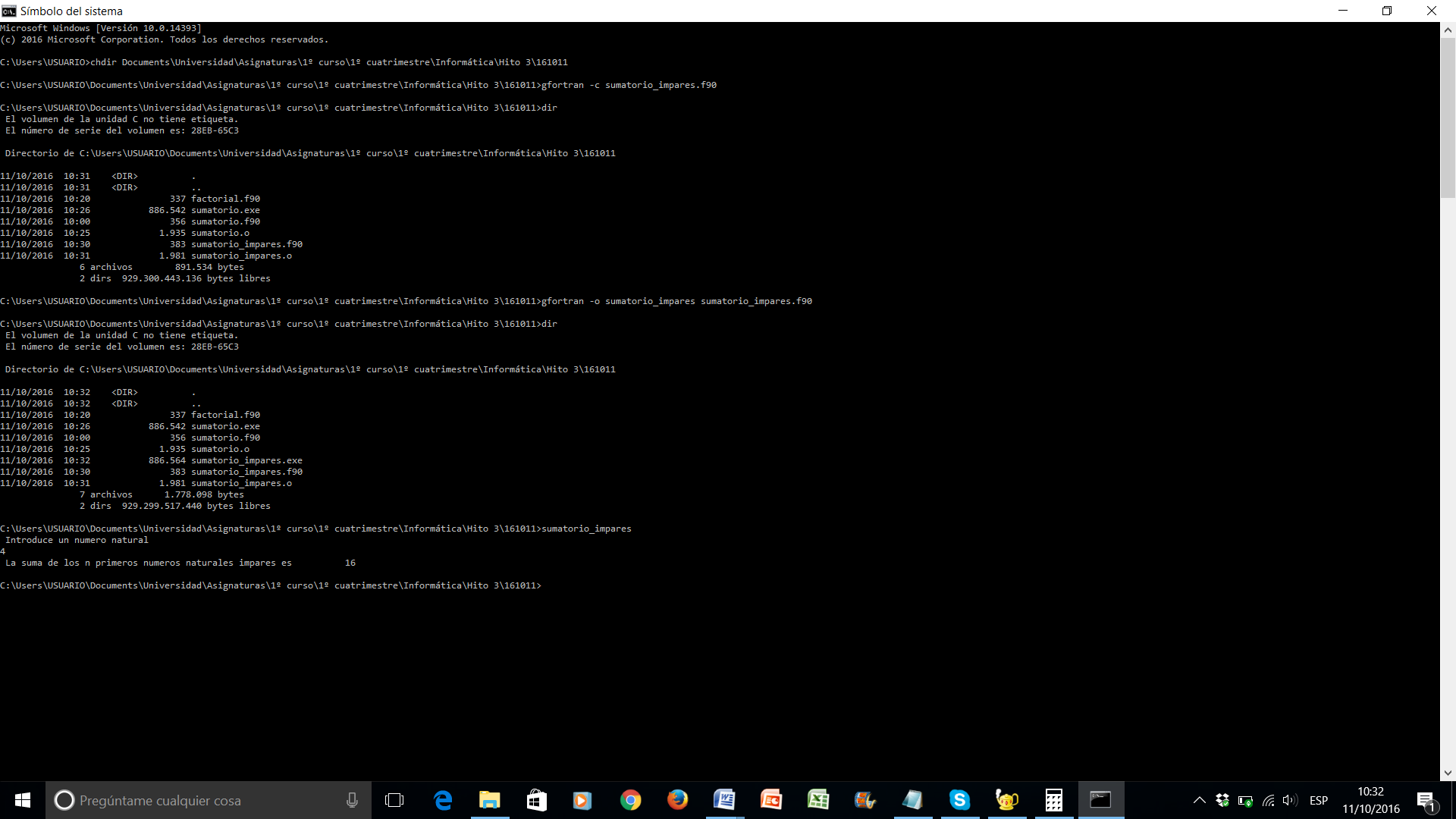
Prueba del programa “sumatorio” en ‘Símbolo del sistema’

*PROGRAMA 2: “sumatorio\_impares”*

Compilación del programa “sumatorio\_impares” en ‘Símbolo del sistema’

Ejecución del programa “sumatorio\_impares” en ‘Símbolo del sistema’



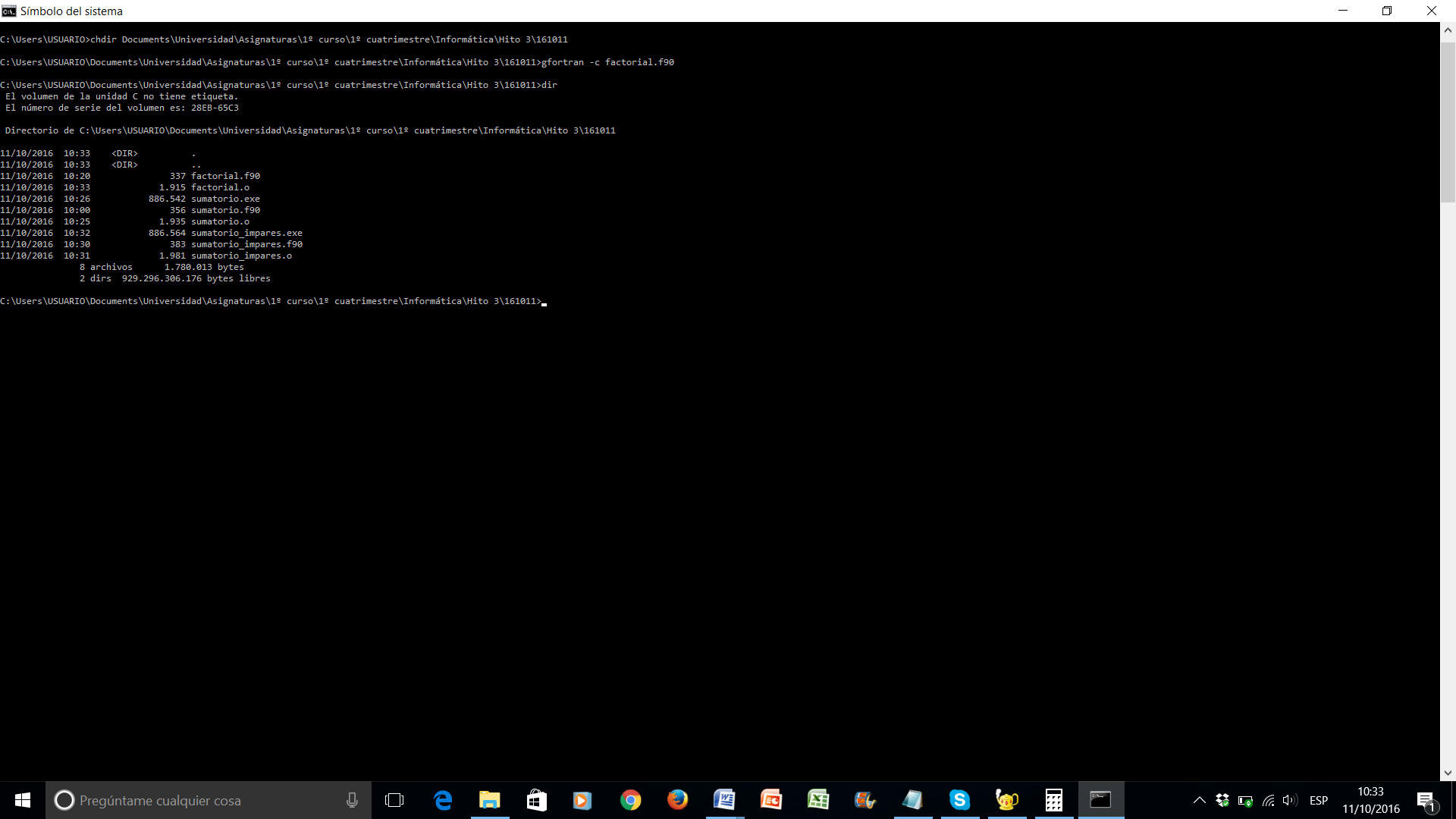
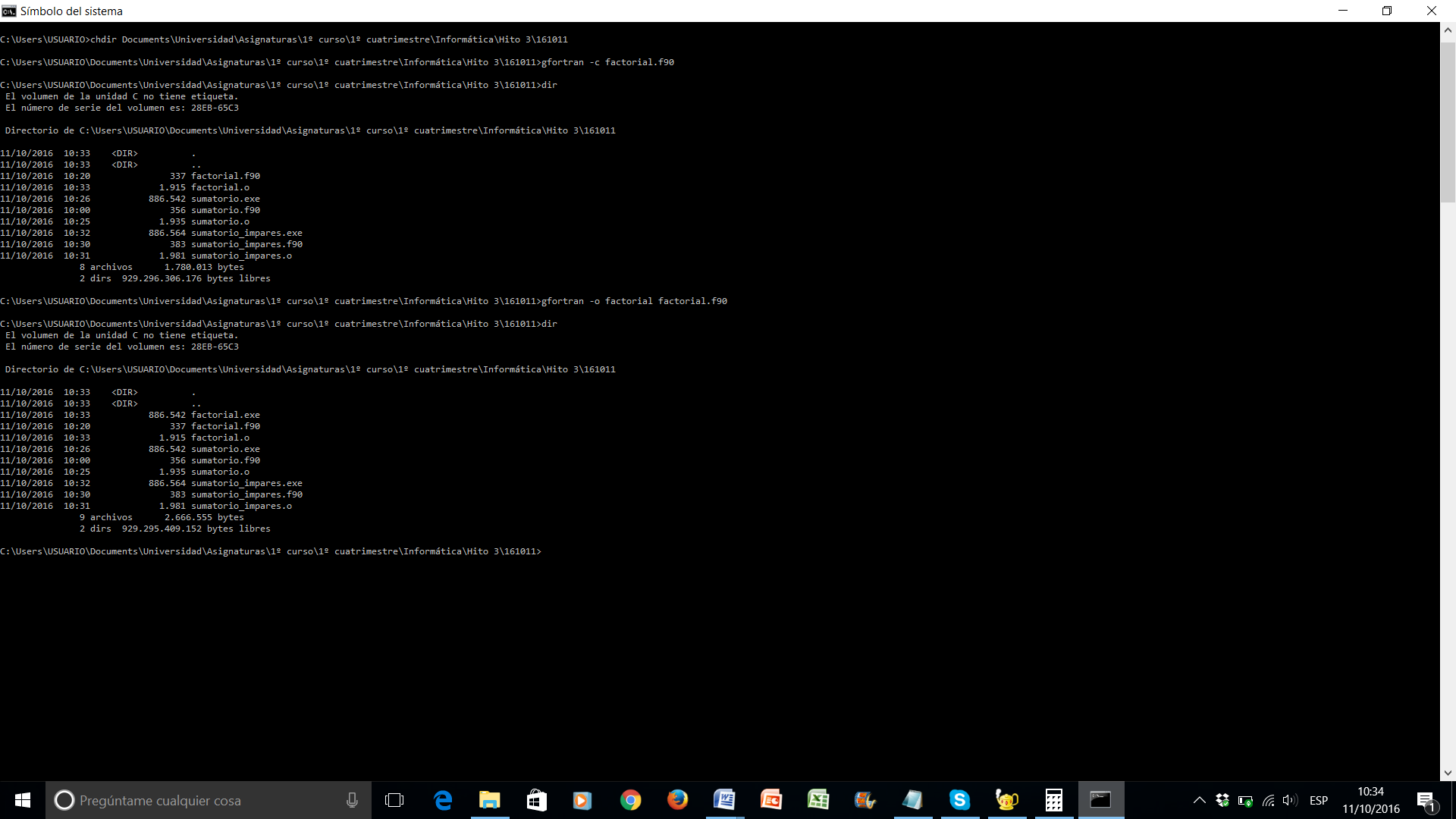


Escritura del programa “sumatorio\_impares” en el editor *Geany*

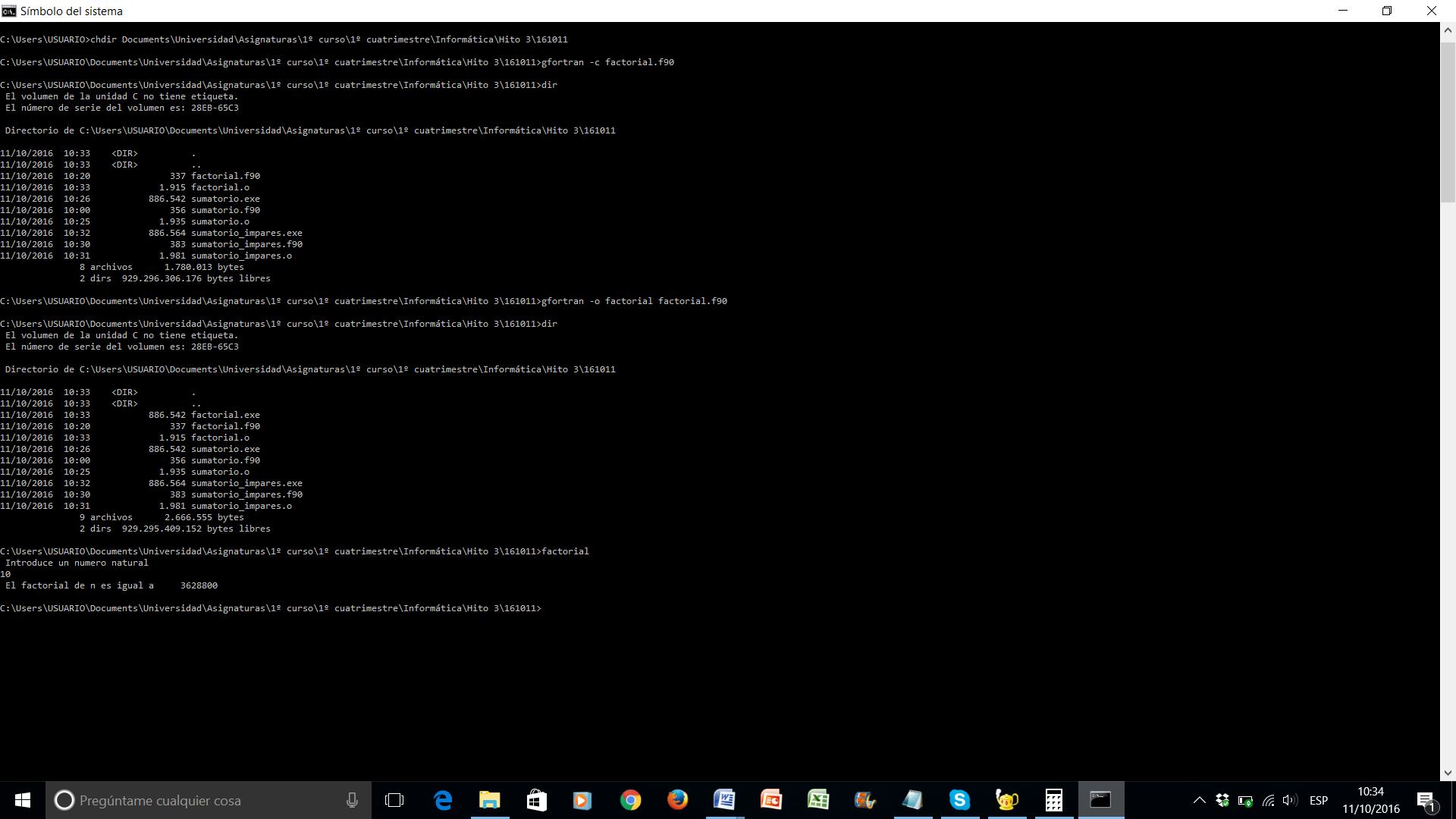
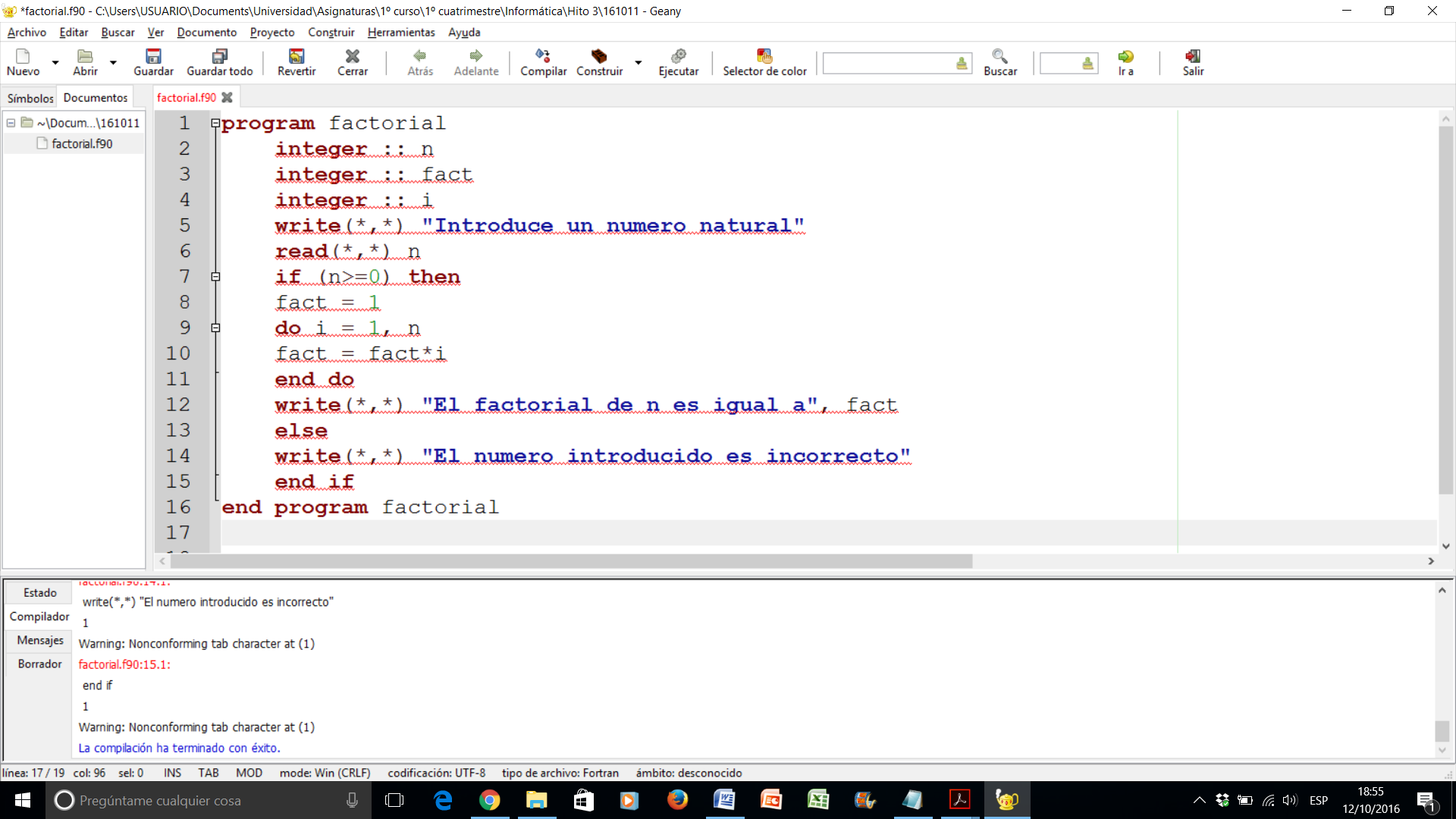
Prueba del programa “sumatorio\_impares” en ‘Símbolo del sistema’

*PROGRAMA 3: “factorial”*

Compilación del programa “factorial” en ‘Símbolo del sistema’

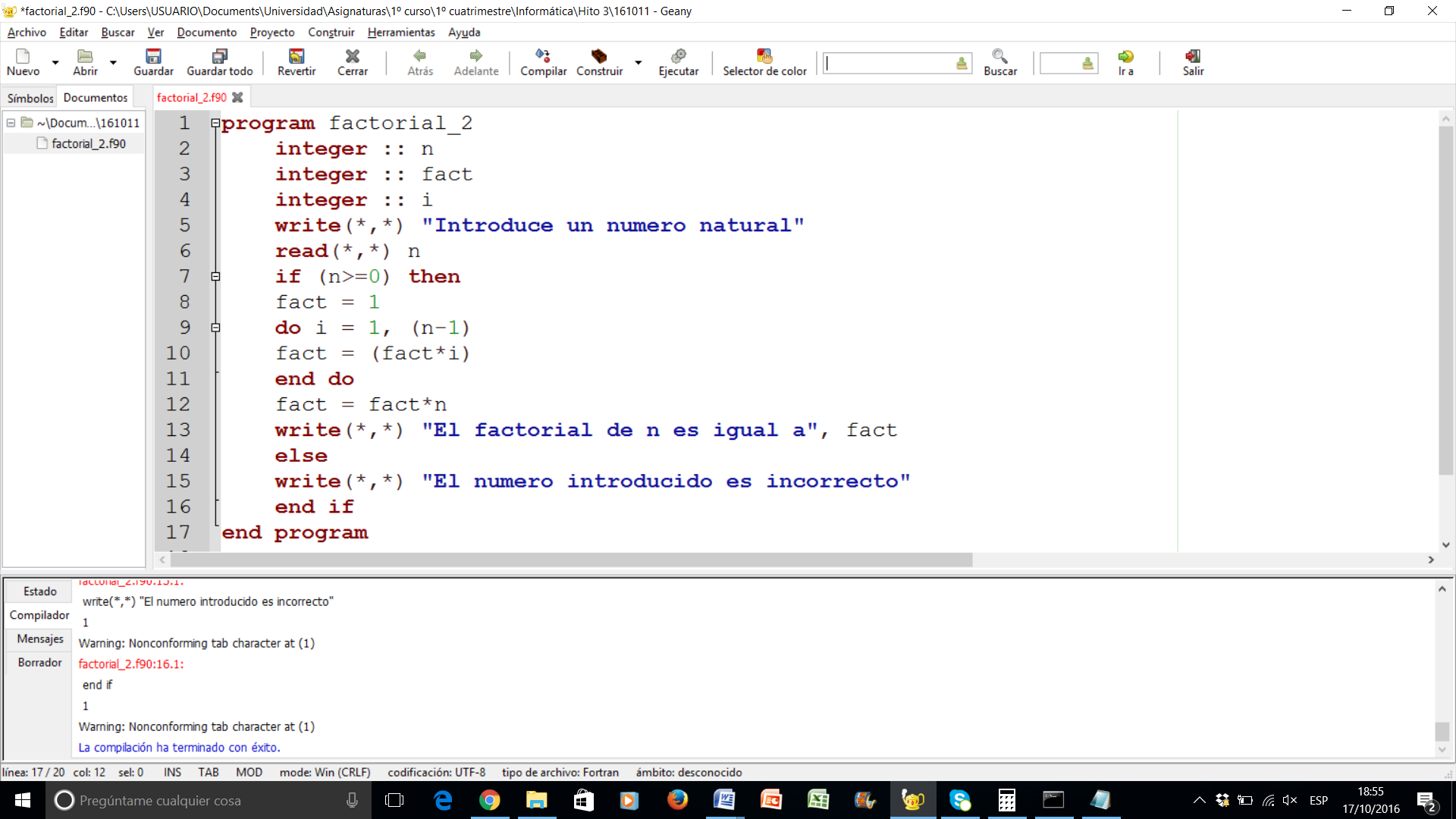


Ejecución del programa “factorial” en ‘Símbolo del sistema’



Escritura del programa “factorial” en el editor *Geany*

Prueba del programa “factorial” en ‘Símbolo del sistema’



Otra forma de escribir el programa “factorial” en *Geany*

**LIMITACIONES NUMÉRICAS LA VARIABLE ENTERA EN “factorial”**

Como se decía en la introducción teórica, la imagen de la función factorial crece muy rápidamente. A la variable entera se le pueden asignar en Fortran 90 distintos identificadores de clase: 1, 2, 4 y hasta 8 *bytes*. Este valor supondrá el espacio numérico máximo que podrá soportar la variable “fact” (fact = n!).

Solamente hemos trabajado en el programa con los primeros tres identificadores de clase:

- kind = 1 → 1 byte ≡ 8 bits → 28 – 1 = 255 → (-128, 127) → **máximo valor de “n” válido: 5** → 5! = 120 < 127, pues 6! = 720 > 127.

- kind = 2 → 2 bytes ≡ 16 bits → 216 – 1 = 65 535 → (-32 768, 32 767) → **máximo valor de “n” válido: 7** → 7! = 5 040 < 32 767, pues 8! = 40 320 > 32 767.

- kind = 4 → 4 bytes ≡ 32 bits → 232 – 1 = 4 294 967 295 → (-2 147 483 648, 2 147 483 647) → **máximo valor de “n” válido: 12** → 12! = 479 001 600 < 2 147 483 647, pues 13! = 6 227 020 800 > 2 147 483 647.

**CONCLUSIÓN**

Trabajar esta vez con el nuevo “accesorio”, el bucle “do”, no ha resultado especialmente costoso. Así como al introducir en el hito anterior la sentencia “if” sí tuvimos ciertos problemas, los solucionamos fácilmente en este al tener que reintroducirla.

Creemos ahora saber manejar decentemente el bucle “do” en aplicaciones similares a las de los programas realizados: series de sumatorios, de productorios, etc. También hemos podido comprobar las “limitaciones” numéricas de la variable entera, que se dejan entrever sobre todo en el último programa.

Por último, cabe decir que ya hemos interiorizado el proceso de compilación y ejecución en la consola, que se ha convertido ya en un mero trámite.

*Evaristo de Vega Galindo Yag*o *Pego Martínez*