TEMA 2: Elementos básicos de programación

1. Notación BFN

Un lenguaje de programación sigue unas reglas gramaticales similares a las de cualquier idioma humano, aunque más estrictas. Para la definición formal de dichas reglas sintácticas utilizaremos la notación BNF(Backus-Naur Form):

* METASIMBOLOS

|  |  |
| --- | --- |
| **::=** | Definición: Indica que el elemento a su izquierda puede desarrollarse según el esquema de la derecha. |
| **|** | Alternativa. Indica que puede elegirse uno y solo uno de los elementos separados por este meta-símbolo. |
| **{ }** | Repetición. Lo elementos dentro de ella se pueden repetir cero o más veces. |
| **[ ]** | Opción. Indica que los elementos de ellos pueden ser utilizados o no. |
| **( )** | Agrupación. Agrupan los elementos incluidos en su interior. |

Los elementos no terminales y terminales se escribirán con el siguiente formato:

*Elemento\_no\_terminal ::=* **Elemento\_terminal**

1. **Valores y tipos.**

Definamos dato como un elemento de información que puede tomar un valor entre varios posibles. Si un dato siempre tiene el mismo el valor fijo, diremos que es una constante.

TIPOS ABSTRACTOS DE DATOS

Los tipos abstractos de datos sirven para identificar el conjunto de valores que pueden tomar los datos de ese tipo como las operaciones que se pueden hacer con dichos datos.

**Valores constantes:** Son valores cuyo valor permanece inalterado durante su uso.

**Valores numéricos enteros**: estos números representan un número exacto de unidades, y no pueden tener parte decimal. Pueden estar precedidos opcionalmente por los símbolos (+) o (-).

**Valores numéricos reales**: Los valores numéricos reales permiten representar cualquier cantidad, naturales, enteros, racionales, fraccionarios…etc. Normalmente lo que delimita la diferencia entre parte real y parte entera es un punto “.”

**Caracteres:** representan los caracteres propios de un texto, que están disponible en cualquier teclado, pantalla o impresora. En un programa de C+- el valor de un carácter concreto se escribe poniendo dicho carácter entre apostrofes (‘). Otros caracteres definidos, como los ***caracteres de control***, que no tienen símbolo grafico se representan mediante una secuencia de escape.

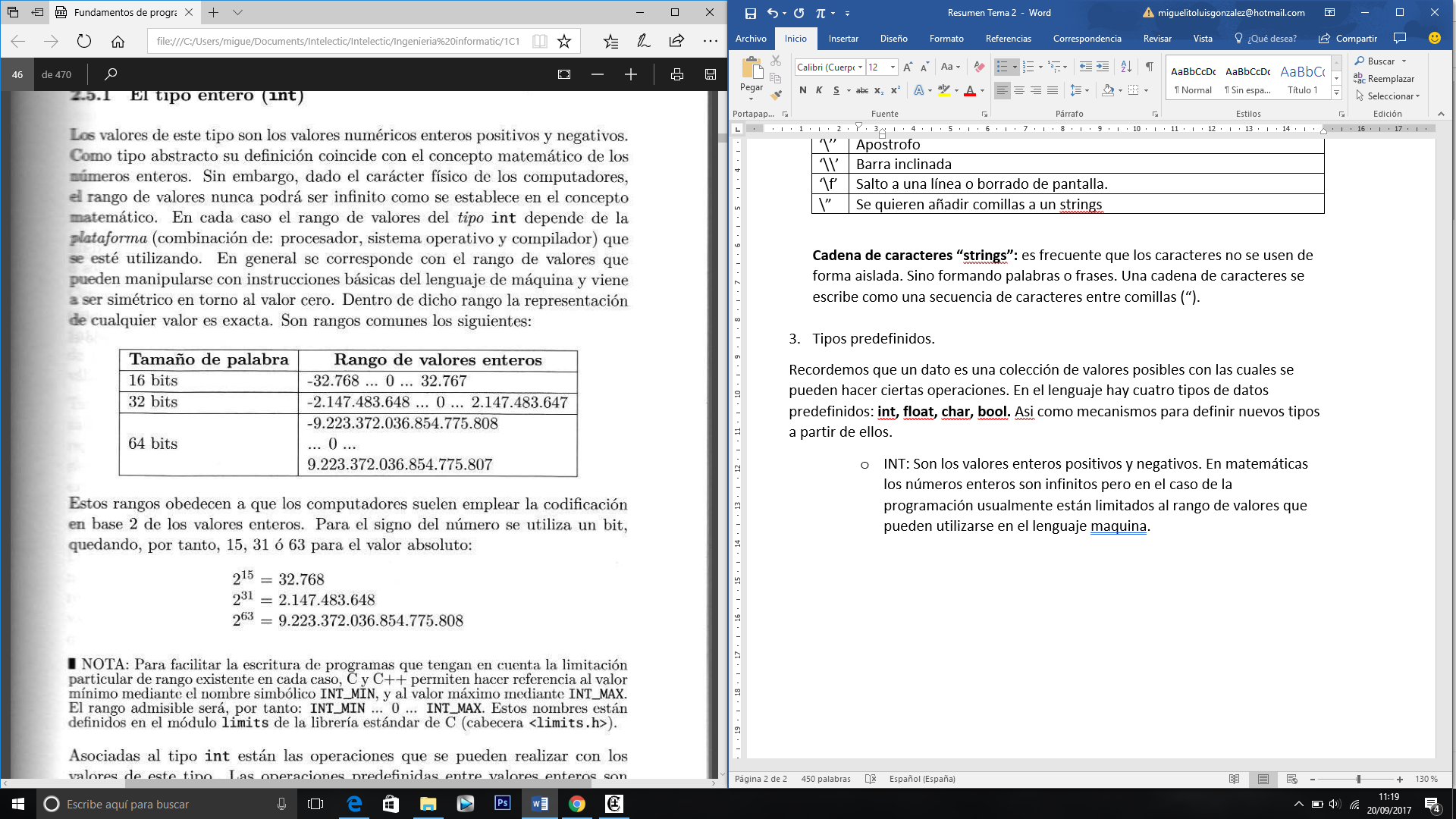
|  |  |
| --- | --- |
| ‘\n’ | Salto al comienzo de una nueva línea de escritura |
| ‘\r’ | Retorno al comienzo de la misma line de escritura |
| ‘\t’ | Tabulación |
| ‘\’’ | Apostrofo |
| ‘\\’ | Barra inclinada |
| ‘\f’ | Salto a una línea o borrado de pantalla. |
| \” | Se quieren añadir comillas a un strings |

**Cadena de caracteres “strings”:** es frecuente que los caracteres no se usen de forma aislada. Sino formando palabras o frases. Una cadena de caracteres se escribe como una secuencia de caracteres entre comillas (“).

1. **Tipos predefinidos.**

Recordemos que un dato es una colección de valores posibles con las cuales se pueden hacer ciertas operaciones. En el lenguaje hay cuatro tipos de datos predefinidos: **int, float, char, bool.** Así como mecanismos para definir nuevos tipos a partir de ellos.

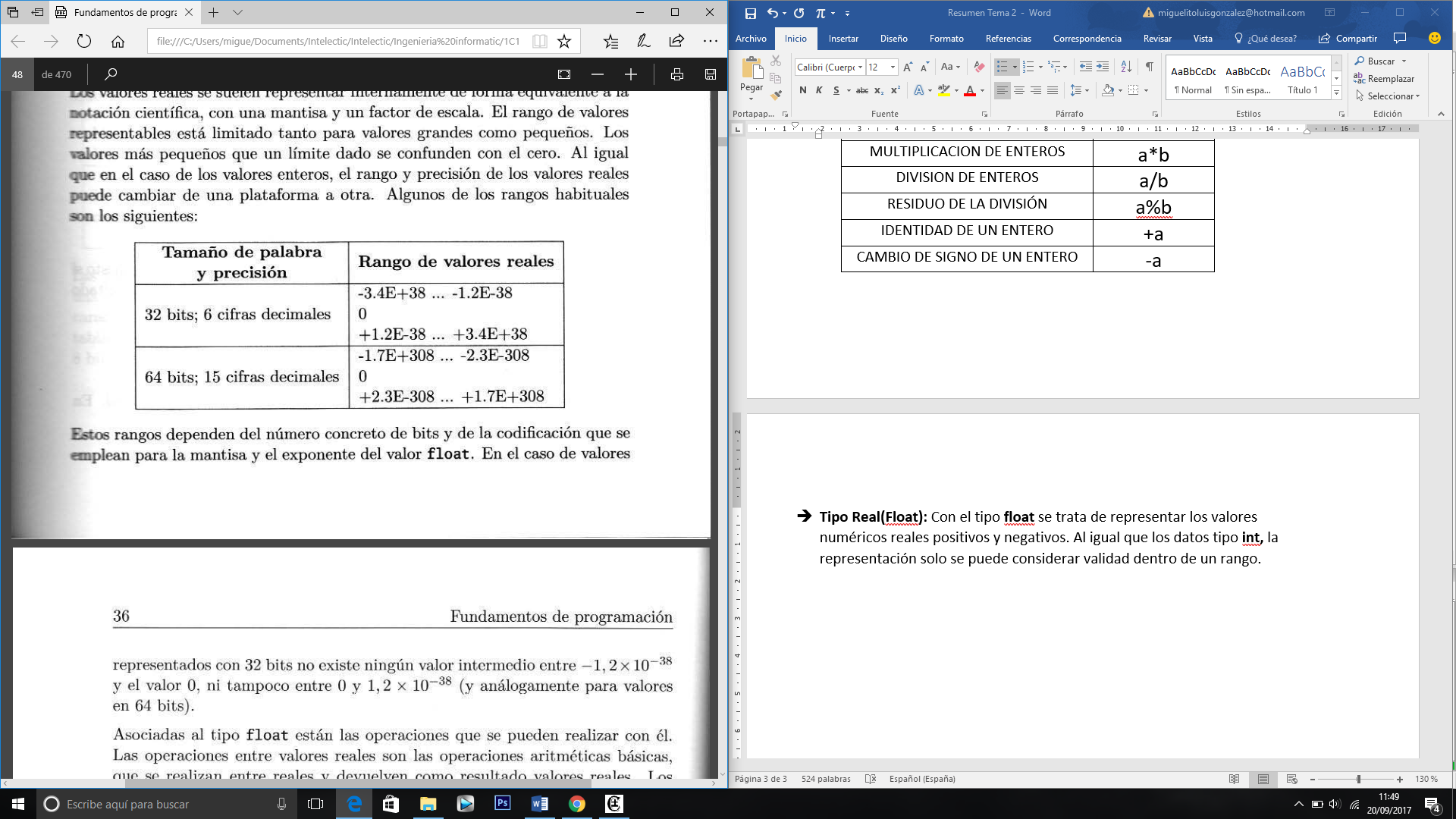
* **INT:** Son los valores enteros positivos y negativos. En matemáticas los números enteros son infinitos, pero en el caso de la programación usualmente están limitados al rango de valores que pueden utilizarse en el lenguaje máquina.



# OPERACIONES CON DATOS TIPO INT

|  |  |
| --- | --- |
| SUMA DE ENTEROS | a+b |
| RESTA DE ENTEROS | a-b |
| MULTIPLICACION DE ENTEROS | a\*b |
| DIVISION DE ENTEROS | a/b |
| RESIDUO DE LA DIVISIÓN | a%b |
| IDENTIDAD DE UN ENTERO | +a |
| CAMBIO DE SIGNO DE UN ENTERO | -a |

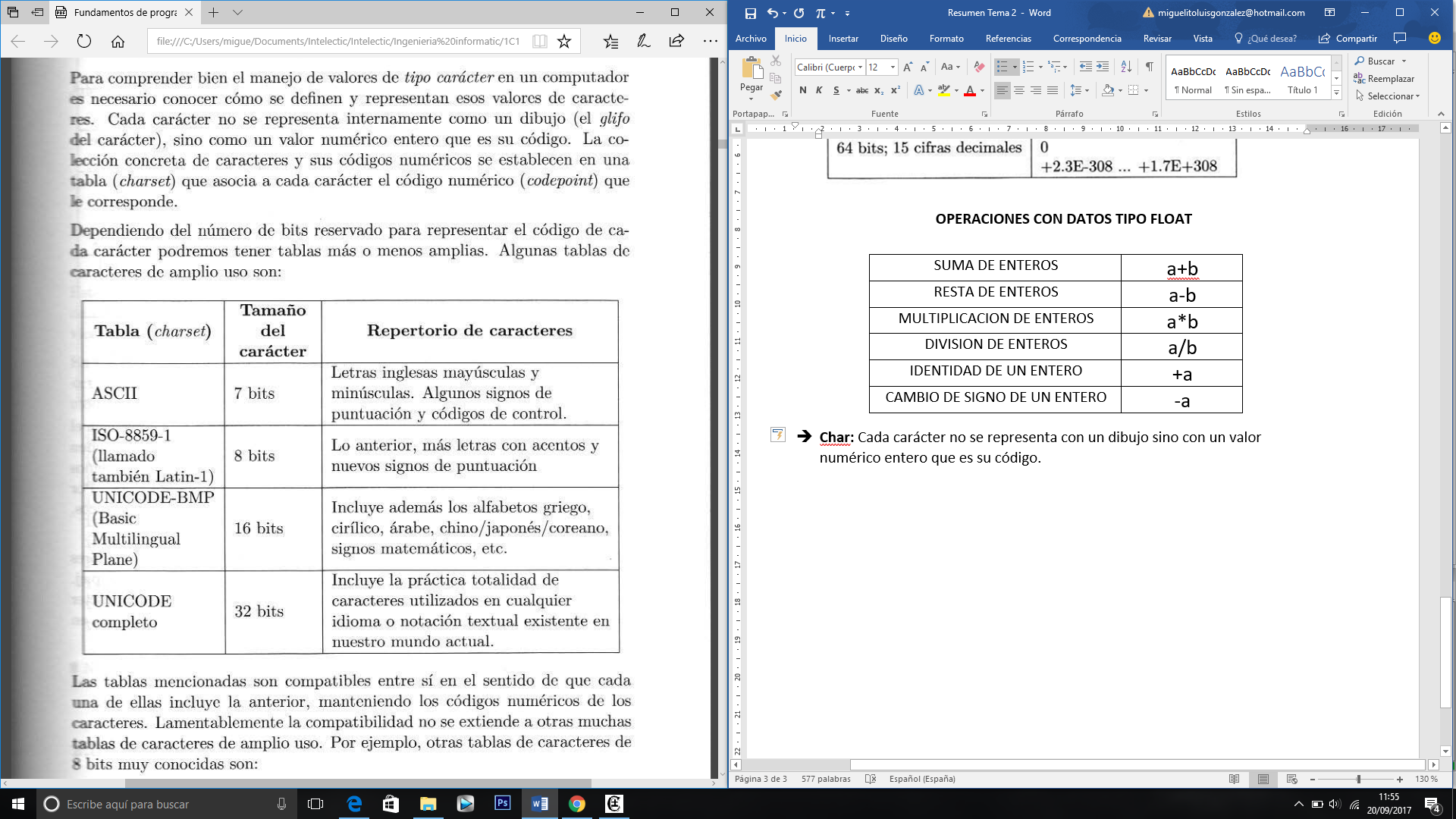
* **Tipo Real(Float):** Con el tipo **float** se trata de representar los valores numéricos reales positivos y negativos. Al igual que los datos tipo **int,** la representación solo se puede considerar validad dentro de un rango.



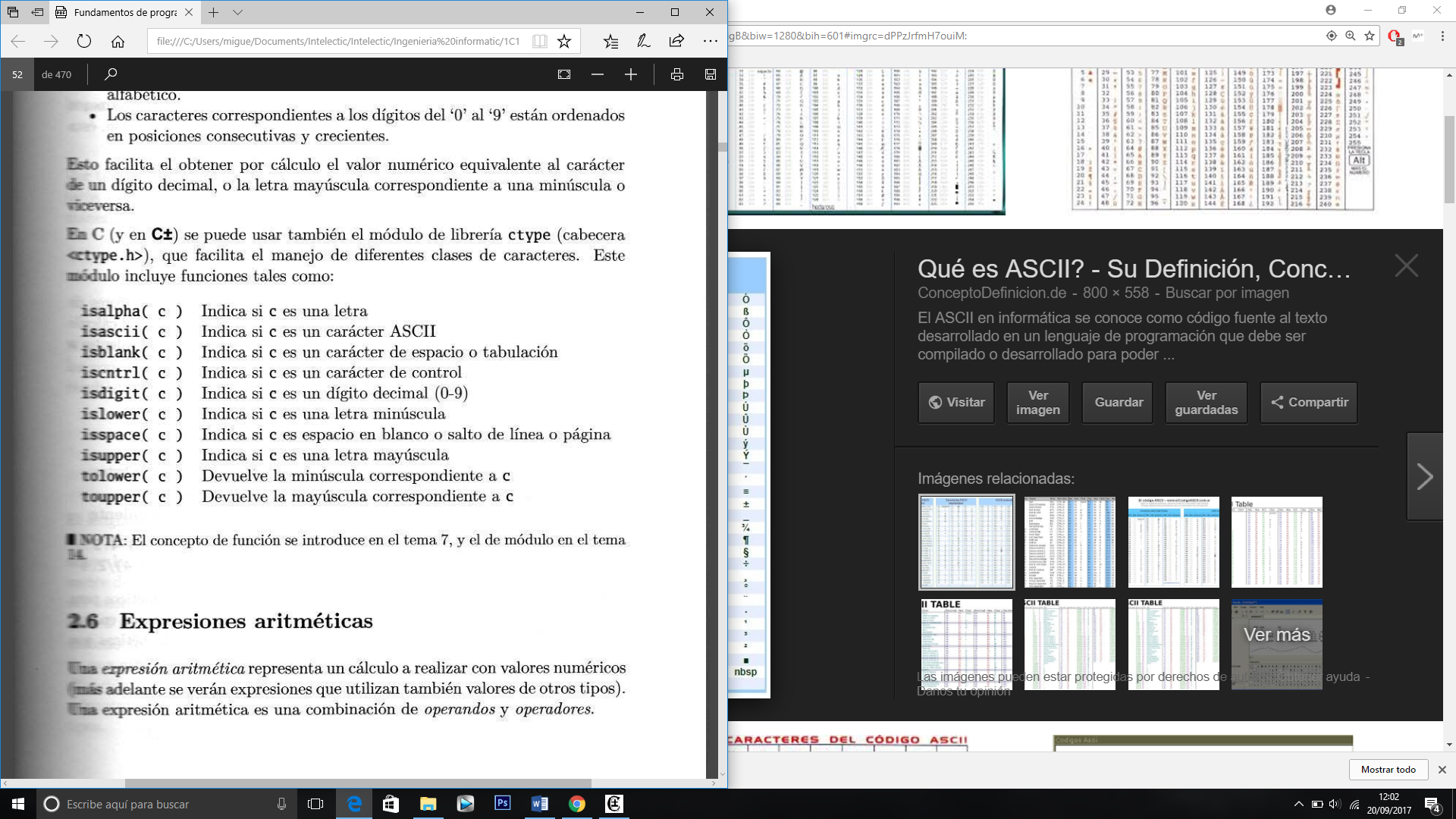
**OPERACIONES CON DATOS TIPO FLOAT**

|  |  |
| --- | --- |
| SUMA DE ENTEROS | a+b |
| RESTA DE ENTEROS | a-b |
| MULTIPLICACION DE ENTEROS | a\*b |
| DIVISION DE ENTEROS | a/b |
| IDENTIDAD DE UN ENTERO | +a |
| CAMBIO DE SIGNO DE UN ENTERO | -a |

* **Char:** Cada carácter no se representa con un dibujo sino con un valor numérico entero que es su código. Charset es la colección concreta de caracteres, el tamaño del carácter es la cantidad de bits que usa para representar cada carácter. El leguaje de programación c+- lo valores del tipo char ocupan 8 bits e incluye el repertorio ASCCI.



Si incluimos la librería <ctype.h> este modulo incluye funciones tales como:



1. **Expresiones aritméticas.**

Una expresión aritmética representa un cálculo a realizar con valores numéricos. Es una combinación de operando y operadores. Para indicar el orden en el que se quieren realizar las operaciones parciales se pueden utilizar paréntesis.

* float(45) = convierte el numero 45 entero en un numero 45 real.
  + Ej. Float(45)= 45.0
* int(34,7) = convierte el numero 34,7 real en un numero entero
  + EJ. Int(34.7)= 37
* Char(168) = devuelve un carácter correspondiente al número 168 en el código ASCCI, en este caso ‘¿’

1. **Operaciones de escritura simples.**

El objetivo de un programa es obtener unos resultados. Estos resultados debenser emitidos al exterior del computador a través de un dispositivo de salida de datos: impresora, pantalla, trazador, línea de comunicaciones…etc. Las acciones que se envían al exterior se llaman en general “*operaciones de escritura”*

Existe una gran variedad de dispositivos periféricos, que se diferencian mucho en los detalles de su manejo. Al diseñar un lenguaje de programación se puede optar por usar sentencias o instrucciones especiales para ordenar la escritura de resultados, o bien ordenar la escritura del resultado con las mismas sentencias generales que se emplean para invocar operaciones definidas por el usuario.

* printf: Este procedimiento pertenece al módulo <stdio.h>. La forma más sencilla de invocarlo es escribir.
  + printf(cadena-de-caracteres): esta forma sencilla de escribir solo es validad si la cadena de caracteres a escribir no contiene el carácter “%” . El procedimiento printf escribe en la pantalla del computador la cadena de caracteres.
  + Si lo que se quiere escribir es la representación como texto de una serie de valores de cualquier tipo de los vistos hasta el momento (int, float, char) habrá que usar la forma general de la orden printf.

Printf(cadena-con-formatos, valor1, valor2,valorN); una cadena de caracteres con formatos deberá incluir en su interior una especificación de formato por cada valor que se quiera insertar. La forma mas simple de especificar un formato es mediante %x,, es decir, usando el carácter fijo “%” seguido de una letra de código que indica el tipo de formato a aplicar. Algunos codigos de formatos habituales son:

|  |  |  |
| --- | --- | --- |
| Código | Nemotecnico  (ingles) | Tipo de valor |
| d | **d**ecimal | entero |
| f | **f**ixed point | real |
| e | **e**xponential | Real con notación exponencial |
| g | **g**eneral | Real con/sin notación exponencial |
| c | **c**haracter | Un caracter |
| s | **s**tring | Una cadena de caracteres |

EJEMPLOS:

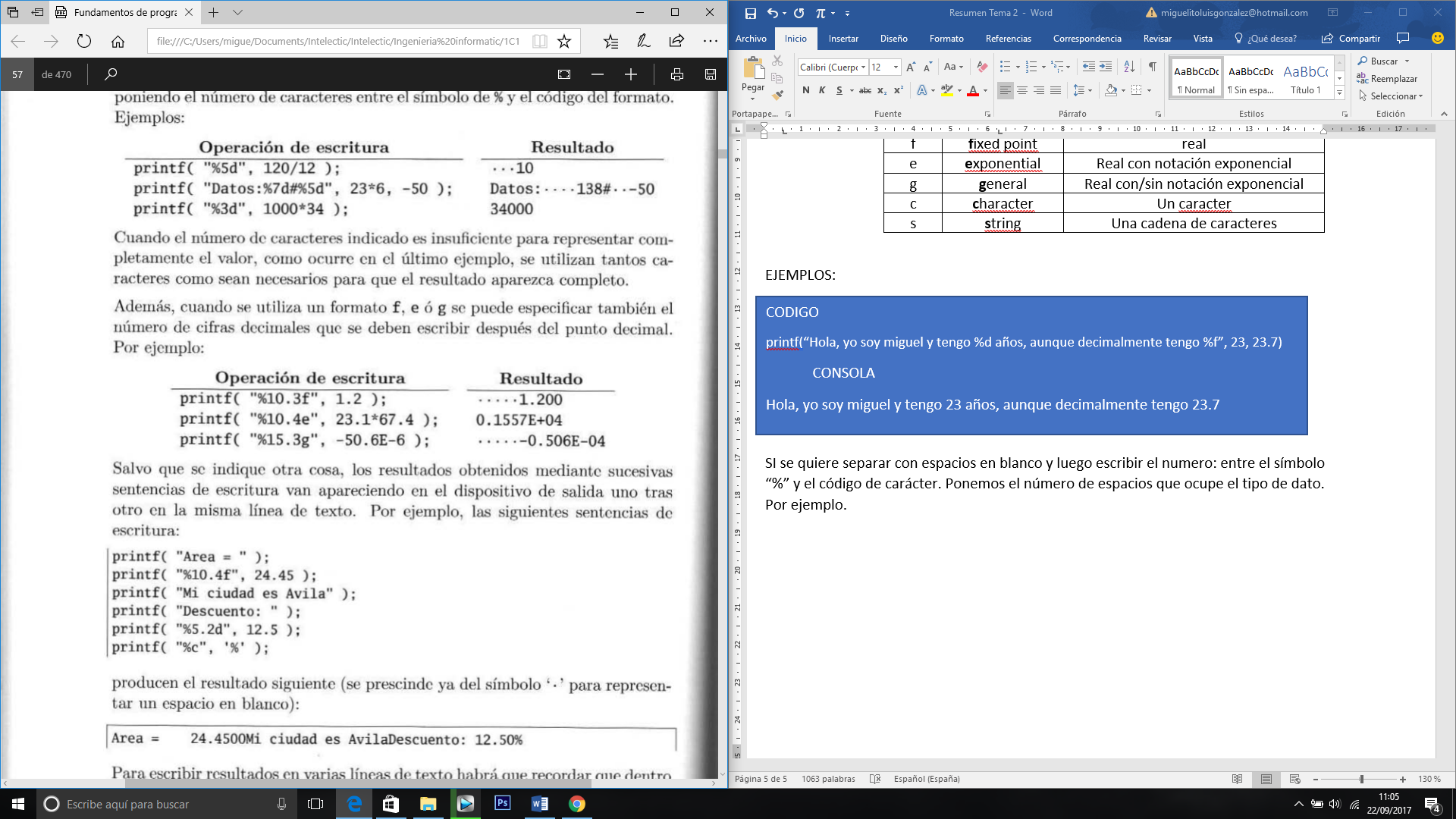
CODIGO

printf(“Hola, yo soy miguel y tengo %d años, aunque decimalmente tengo %f”, 23, 23.7)

CONSOLA

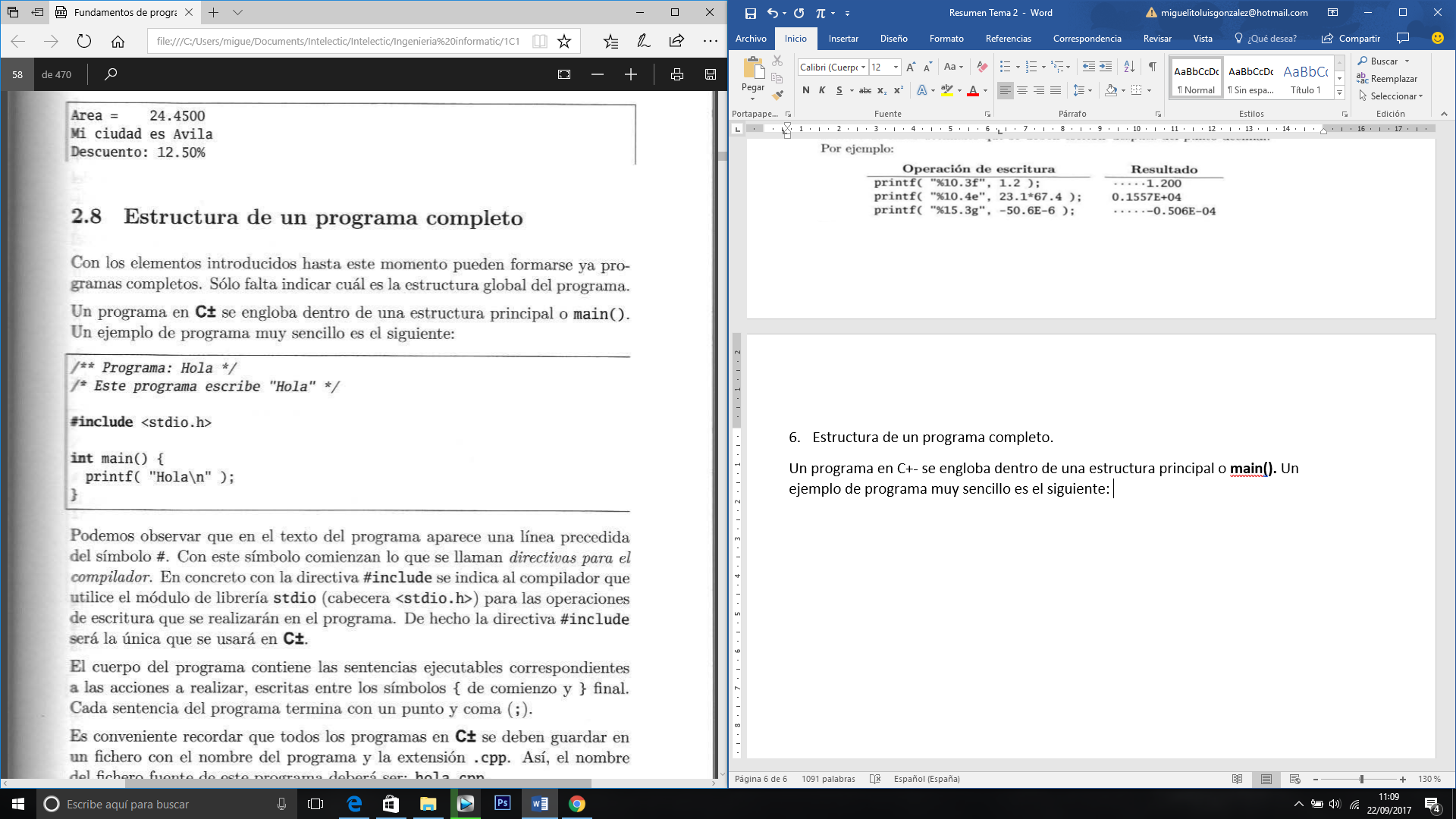
Hola, yo soy miguel y tengo 23 años, aunque decimalmente tengo 23.7

SI se quiere separar con espacios en blanco y luego escribir el numero: entre el símbolo “%” y el código de carácter. Ponemos el número de espacios que ocupe el tipo de dato. Por ejemplo.



1. Estructura de un programa completo.

Un programa en C+- se engloba dentro de una estructura principal o **main().** Un ejemplo de programa muy sencillo es el siguiente:



Con el símbolo “#” comienzan directivas para el compilador. En concreto con la directiva #include se indica al compilador que utilice el módulo de librería stdio. La directiva include será la única que se utilizará en C+-. El cuerpo del programa compuesto por las sentencias ejecutables a realizar esta escrita entre los símbolos{programa} y cada directiva, sentencia o declaración finaliza con el símbolo “;”

1. Uso de comentarios.

La descripción formal de la estructura de un programa simplificada de un programa es la siguiente:

Include::= #include <Nombre\_modulo.h>

Programa ::= {include} int main() Bloque

Bloque ::= {Parte ejecutiva}

Parte\_ejecutiva::= {sentencia}

* Cada directiva puede incluir una línea de programa únicamente.
* Para escribir comentarios utilizamos el símbolo “/\*” para comenzar el comentario y el símbolo “\*/” para finalizar el comentario. Estos comentarios sirven principalmente para recordarnos que es lo que hace cada sentencia…etc.