***Introducción a la computación: Taller Flash N°4***

1. Representación de números con coma

1.1. Punto fijo

La función **fix** devuelve el entero de la multiplicación del valor de entrada por *divisor*.

**fix**(-53)=-53000

**fix**(0.321587)=321

La función **unfix** devuelve el resultado de dividir el valor de entrada por *divisor*.

**unfix**(-53)=-0.053

**unfix**(0.321587)=0.000321587

Si se combinan ambas funciones de la siguiente forma **unfix**(**fix**(value)), podemos representar por punto fijo un determinado valor e ir modificando las posiciones después de la coma representadas cambiando el valor de *divisor*.

En los códigos siguientes se busca el resultado de :

1. (fix(10.05)-fix(15.75))//fix(3.1415)= -2
2. unfix(fix(fix(10.05)-fix(15.75))//fix(3.1415))= -1.815
3. unfix(fix(fix(10.05)-fix(15.75))//fix(3.1415))= -1.81442
4. (10.05-15.75)/3.1415= -1.8144198631227118

En a) no se conserva ningún decimal para informar el resultado, en b) sí, al estar usándose la forma combinada de ambas funciones. Aquí el valor de *divisor*=1000. En c) se modificó la presición a 5 decimales, es decir *divisor*=100000. Por último, en d) se escribió la operación directamente en Python y devuelve un número con 16 posiciones decimales.

1.2. Punto Flotante

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Codificación de los siguientes elementos: | notación fija (precisión a 5 decimales) | tipo de datos *float* de Python | Programación de Pila de C |
| 1 | 0,00000456 | 0.0 | 4.56e-06 | - |
| 2 | 9223372036854775807 | 9.223372036854776e+18 | 9.223372036854776e+18 | - |
| 3 | 43,001 · 0,00751 | 0.32293 | 0.32293751 | 3.229375e-01 |
| 4 | 0,00000001 · 1,4142135623730951 | 0.0 | 1.4142135623730952e-08 | 1.414214e-08 |
| 5 | 0,1 + 0,1 + 0,1 − 0,3 | 0.0 | 5.551115123125783e-17 | 0.000000e+00 |

En el primer caso, la representación por notación fija ‘trunca’ al número en el quinto decimal, la representación por punto flotante no. En el segundo caso el número es un entero entonces ambas representaciones coinciden. Tanto en el tercer como en el cuarto caso, la representación por punto fijo del resultado de la multiplicación trunca el resultado en el quinto decimal mientras que la representación por punto flotante da un resultado más exacto. Por último, como resultado de la suma de números con un decimal, la representación por punto fijo devuelve el resultado real, mientras que la representación por punto flotante da una aproximación (con muy poco error) del resultado.

Es decir la representación por punto fijo no es útil al querer utilizar números por fuera del rango de presición con la que se está operando, pero si se trabaja con números en ese rango, este tipo de representación es útil ya que no aproxima el valor del número si no que devuelve el resultado real y además es una operación que utiliza menos bytes (en este ejemplo, que se utiliza una presición de 5 decimales)

2. Estándar IEEE-754

2.1. Representación

La función **verBits** transforma un número en base decimal a formato IEE-754.

**verBits**(1)= 00111111100000000000000000000000

**verBits**(-1)= 10111111100000000000000000000000

**verBits**(0.5)= 00111111000000000000000000000000

**verBits**(float('inf'))= 01111111100000000000000000000000

**verBits**(float('-inf'))= 11111111100000000000000000000000

**verBits**(float('NaN'))=01111111110000000000000000000000

El primer bit (signo) es 0, porque es un número positivo, el exponente vale 127 porque al restarlo por el exceso es 0 y la mantisa son todos 0 porque está normalizada, quiere decir que el primer 1 no se coloca. Entonces queda

La única diferencia con el número anterior es el 1 en el primer bit que corresponde al signo negativo.

El bit de signo es 0, el exponente vale 126 y la mantisa son todos 0, de manera tal que

NaN (not a number)

Para representar este elemento 0-11111111-mantisa. La mantisa necesita por lo menos un 1.

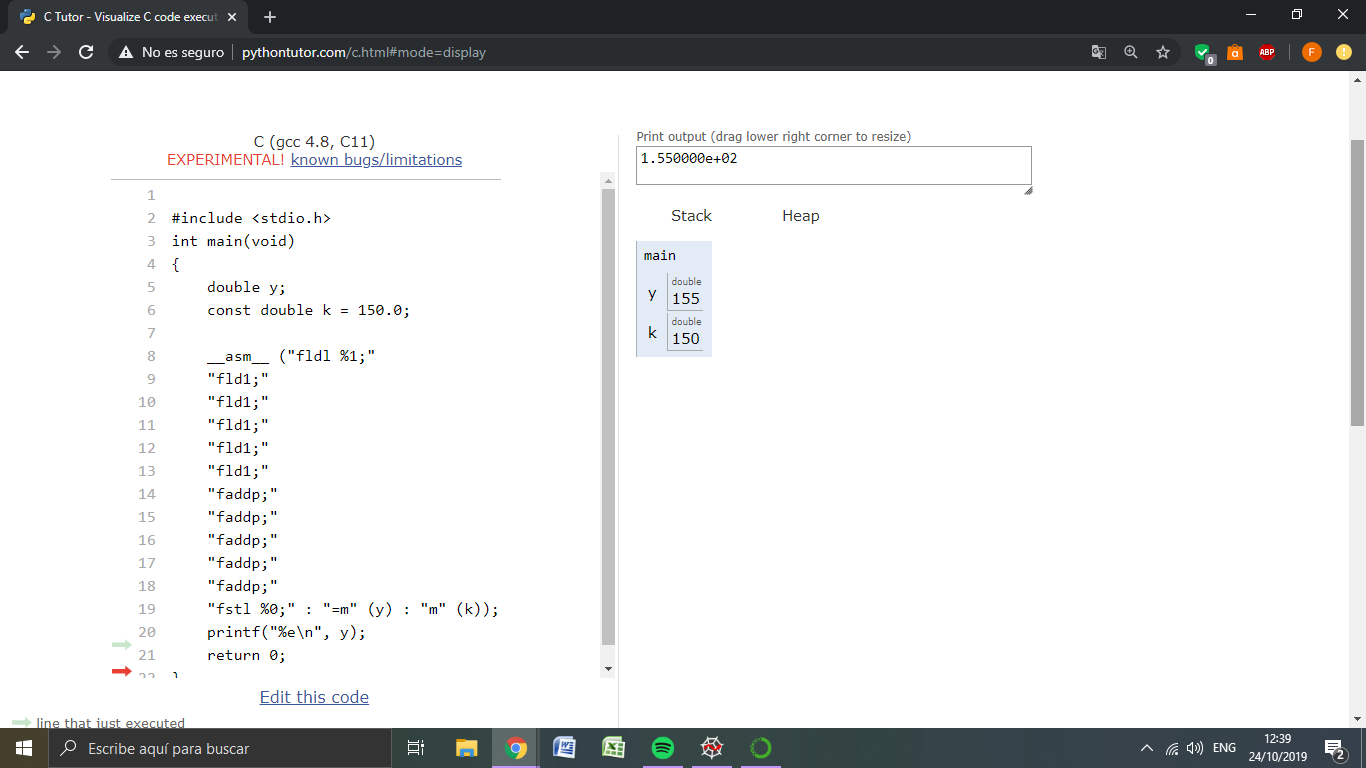
+Infinito

El bit del signo es 0 (si se quiere representar el -infinito alcanza con que el bit valga 1) , el exponente son todos 1 (máximo valor 255-127= 128) y la mantisa son todos 0.

3. Unidad de Punto Flotante

3.1. Programación de Pila

Una forma para sumar 150+5 es repetir los comandos de colocar un 1 en la última pila (*fld1)* y sumar esa posición al 150 cinco veces (*faddp)*, como muestra el siguiente script:



3.1.1. Dos operandos

Modificando los valores de *a* y *b* y el comando *faddp* por *fmulp* con respecto al script de referencia se puede obtener una multiplicación de dos números (ver tabla arriba).