

P3: Divide y vencerás

A la hora de realizar la práctica, como suelo hacer a la hora de programar, empecé con un lápiz y un papel escribiendo qué necesitaría la clase. Como dice el enunciado, necesitaba la sobrecarga de operadores ``<<``, ``>>``, ``*`` y ``+``. Los dos primeros fueron bastante fáciles de programar ya que son los que siempre se suelen sobrecargar. Como extra, busqué una función eficiente para comprobar que el número introducido por el usuario fuese un entero (<http://stackoverflow.com/a/2845275>) y dentro de la sobrecarga del operador de lectura la llamo para solo leer enteros positivos.

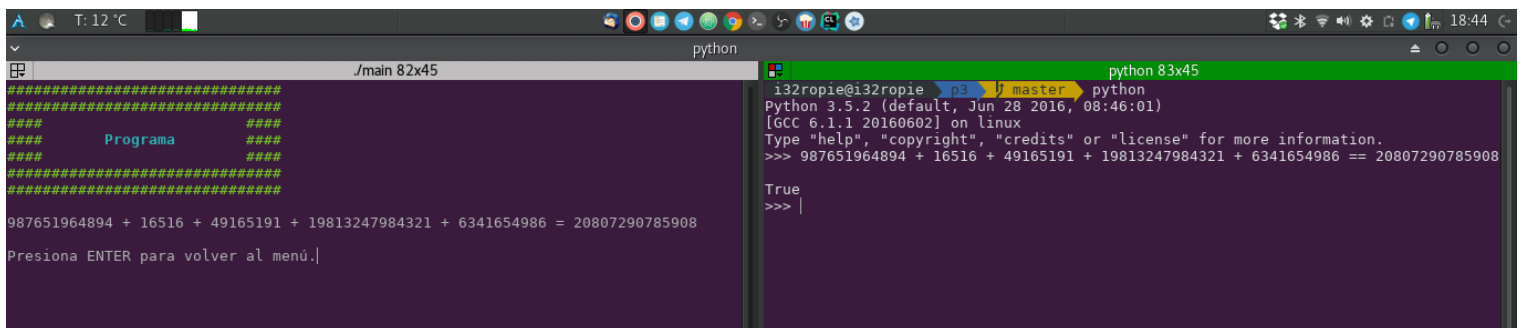
Posteriormente programé la sobrecarga del operador ``+`` y para ello adapté el algoritmo de la suma que aprendemos desde pequeños a C++. Aquí encontré el problema de tener números de distintas cifras, por lo que le añadí 0 no significativos al número menor para hacer las operaciones cómodamente. Para ello, en vez de hacerme una función extra, hice uso de la sobrecarga del constructor del tipo de dato ``std::string`` (<http://www.cplusplus.com/reference/string/string/string/>) que recibe como parámetro un número y un carácter y devuelve una cadena con ese carácter repetido el número de veces que reciba. Mientras hacía esta sobrecarga me encontré con otro problema, y es que yo almaceno en cada iteración en dos ``char`` el valor de la posición *i*-ésima de las cadenas a sumar entonces al intentar transformarlos posteriormente a enteros con la función ``atoi()``, descubrí que al recibir esta como parámetro un ``char *`` necesitaba pasarle la dirección de memoria de los caracteres almacenados y estos se almacenan en la pila uno al lado del otro, de forma decreciente. Esto quiere decir que al hacer ``atoi()`` del segundo ``char``, obtenía los dos caracteres seguidos (<http://cpp.sh/8ti7> Si se ejecuta el código desde la web, ocurre al revés, supongo que tendrá la pila implementada de forma distinta). La solución fue utilizar el método usado en el archivo de ejemplo ``caracteres.c``.

Lo siguiente que implementé a la clase fue la sobrecarga del operador ``*`` que fue bastante sencillo ya que siguiendo el algoritmo visto en clase no tenía mucha pérdida. El algoritmo requiere que el número sea dividido en dos partes cuando el número de cifras del mismo sea superior a un margen. El margen se puede establecer con una función o usar el que la clase establece por defecto, 4. En caso de que el usuario de la clase intente indicar un margen superior a 4, se establecerá automáticamente 4 para asegurar el buen funcionamiento. Respecto a dividir el número en 2 partes, escribí dos sencillas funciones que hacen uso de la función ``substr()``, una para obtener la primera mitad y otra para obtener la segunda. Por último, para generar los exponentes sigo el mismo procedimiento para generar los ceros que cuando añado ceros para hacer que ambos números tengan el mismo número de cifras.

Por último, añadí la sobrecarga de algunos operadores más para dar más facilidades a la hora de usar la clase tales como ``+=``, ``*=`` e ``=`. Este último tiene dos sobrecargas, una que acepta un Entero y otra que acepta una cadena.

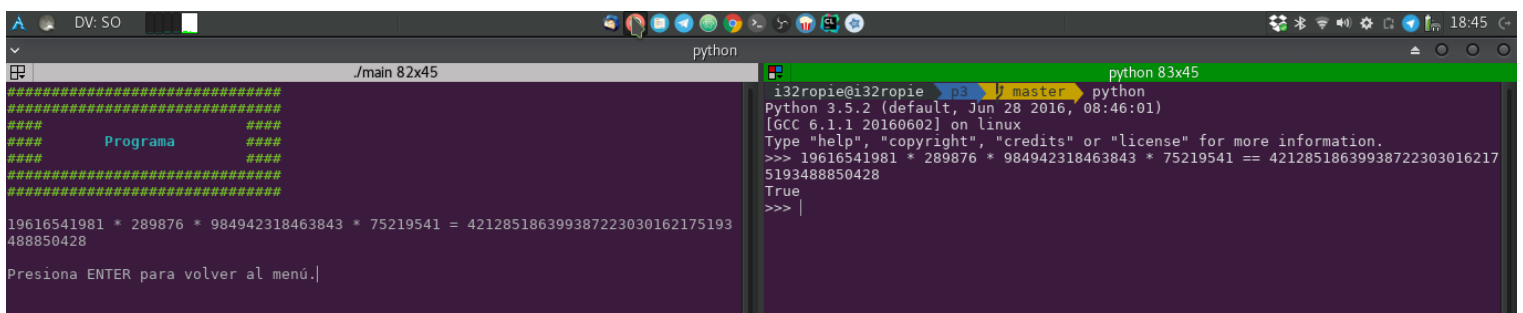
Adicionalmente documenté la práctica con Doxygen, añadí algún comentario aclaratorio dentro del código y en el menú añadí que el usuario del programa sea el que decida cuantos números quiere sumar o multiplicar.

Para comprobar que la clase funciona correctamente, realicé diversas pruebas que fui comprobando con la ayuda de Python ya que gracias a su intérprete, su cómoda sintaxis y su capacidad nativa de operaciones con números grandes facilitó mucho la tarea de comprobación. Adjunto en el documento un par de capturas donde se ve el buen funcionamiento de la práctica.



```
python
./main 82x45
#####
#####
##### Programa #####
#####
#####
987651964894 + 16516 + 49165191 + 19813247984321 + 6341654986 = 20807290785908
Presiona ENTER para volver al menú.]

python 83x45
i32ropie@i32ropie p3 ~/master python
Python 3.5.2 (default, Jun 28 2016, 08:46:01)
[GCC 6.1.1 20160602] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 987651964894 + 16516 + 49165191 + 19813247984321 + 6341654986 == 20807290785908
True
>>> |
```



```
python
./main 82x45
#####
#####
##### Programa #####
#####
#####
19616541981 * 289876 * 984942318463843 * 75219541 = 421285186399387223030162175193
488850428
Presiona ENTER para volver al menú.]

python 83x45
i32ropie@i32ropie p3 ~/master python
Python 3.5.2 (default, Jun 28 2016, 08:46:01)
[GCC 6.1.1 20160602] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 19616541981 * 289876 * 984942318463843 * 75219541 == 42128518639938722303016217
5193488850428
True
>>> |
```