1. Modo Interprete

1.1. Clases

1.1.1. Interprete

Pensé en hacer una clase interprete que se encargue de ir leyendo el archivo brainfuck y lo vaya interpretando a medida que lo vaya leyendo, es decir, no carga en memoria todo el archivo brainfuck. El interprete lee el archivo 2 veces.

La primera vez es para verificar la sintaxis del código. Lo que hay que verificar en este lenguaje, es que los corchetes estén balanceados. Este problema se resolvió de forma muy sencilla con una variable que suma 1 si encuentra un '[' y resta si ']'. Si la variable en algún momento es negativa seguro hay corchetes des-balanceados. Luego los corchetes solo están balanceados si al final la variable tiene el valor 0.

En cuanto la segunda vez que se lee el archivo, lo que hago es leer de a un carácter y según el carácter hago lo que corresponda. En caso de no ser un corchete ('[' o ']') ya puedo hacer algo inmediatamente, por ejemplo '$<$' significa mover el puntero de datos a la izquierda, o '+' significa sumar 1 en la memoria de datos. Entonces los caracteres que se interpretan inmediatamente son: '$<$', '$>$', '+', '-', ',' y '.'. En cuanto los corchetes, es necesario hacer otra cosa. Cuando leo un corchete que abre ('[') tengo que quedarme con todo lo que haya hasta el corchete que cierra (correspondiente) inclusive. Esto se debe a que puede que tenga que ejecutar varias veces todo lo que esta allí. Para hacer esto utilizo por un lado una variable para saber cuando llego al corchete que cierra correspondiente, haciendo algo similar a lo mencionado anteriormente para la primer leída. Además es necesario tener también la información de los saltos, puede pasar que adentro haya otros corchetes (como un while adentro de otro while). Entonces lo que hago es guardarme las posiciones a donde tengo que saltar (en caso de que sea necesario hacerlo). Para esto utilizo una pila donde voy agregando las posiciones de los corchetes que abren y luego desapilo cada vez que encuentro un corchete que cierra. Al desapilar entonces tengo la posición del corchete que abre y la posición del corchete que cierra (correspondientes). Luego esto lo guardo en un diccionario al cual después le puedo preguntar a donde tengo que saltar si estoy en tal posición. De esta forma resuelvo el problema de encontrarme con un while y/o whiles adentro de un while en el código brainfuck.

1.1.2. Memoria de Datos

Utilice una clase para abstraerme de como esta implementada la memoria de datos. El interprete es quien utiliza la misma, ya sea para escribir en esta, aumentar o disminuir en 1 el valor de una celda, mover la posición del puntero de datos, o leer un dato (para luego probablemente imprimirlo por salida estandar). En cuanto la implementación decidí usar un char* debido a que el tamaño de la memoria es fijo, sino viese optado por un vector. Para el puntero de datos use un numero entero, ya que me era mas cómodo.

2. Aclaraciones

Debido a la reentrega del TP1, y los cambios que tuve que hacer, no pude avanzar mucho mas. Estuve haciendo el tutorial de threads y pienso terminar con la 2da parte en los próximos días.