Text

Description automatically generated

Voy por: hacer la documentación; generar código y MV para las funciones especiales; aumentar tamaño de arreglos; hacer que se puedan usar números negativos

**Generación de código y MV para una parte de la aplicación particular**

*Media / Mediana / Moda / Varianza /Desviación estándar / Histograma / Diagrama de caja*

Todas regresan un float menos las funciones gráficas, esas no regresan nada

Los quads serían así: SPECIAL matrix null result

HISTOGRAM matrix null null

BOXPLOT matrix null null

Definir los comandos en el léxico

Hacer la regla en la sintaxis

Añadir la lógica a process\_exp /process\_statement

Añadir la lógica en generate\_quad

Poner la regla de SPECIAL en donde va

Actualizar la propuesta con las nuevas reglas y tokens

Añadir lo que hará la MV

Pruebas para funciones especiales:

1. Avance7\_s [funciones especiales en módulos, aplicadas a arreglos de ints y de floats, todo bien] *pasa*
2. Avance7\_s2 [asignando una func especial a un int, debe dar error] *pasa*

Pruebas:

1. Avance6\_1 *pasa*
2. Avance6\_3 *pasa*
3. Avance6\_6 *pasa*
4. Avance7\_1 [ accesar index negativo] *pasa pero porque no consideré negativos al definir mi léxico…*
5. Avance7\_2 [accesar index fuera de rango] *pasa*
6. Avance3\_3 [de todo, incluido llamadas a funciones; suma de dos llamadas distintas a la misma función; dec y acceso a arreglos] *pasa*
7. factorial [cíclico y recursivo] *pasa*
8. Fibonacci [cíclico y recursivo] *pasa*
9. Vector.txt [find y sort de vectores, multiplicación de matrices]: *pasa*

**Máquina virtual: Ejecución de módulos y arreglos**

*Arreglos*

Si dirección está en el rango de direcciones de pointers (3666-3999), convertirla a la dirección real y sacar el valor

*Módulos*

*Text

Description automatically generated*

*Graphical user interface, text, application, email

Description automatically generated*

**Graphical user interface, text, application

Description automatically generated**

Crear un execution\_stack donde vaya guardando los contextos creados por llamadas a funciones.

En cada iteración del while se debe checar si el stack está vacío. Si no está vacío, se cambia current\_memory

Asimismo crear un ip\_stack donde se guarden las posiciones de los instruction pointers.

ERA:

* Crear instancia de Memory con la info de l\_op
* Añadir instancia a execution\_stack

PARAMETER:

GOSUB:

* Guardar current\_memory en temp
* Cambiar current\_memory a la de execution\_stack.pop()
* Append temp a execution\_stack
* Append instruction\_pointer+1 a ip\_stack
* Instruction\_pointer = rightop virtual address-1

ENDFUNC:

* Current\_memory = execution\_stack.pop()
* Instruction\_pointer = ip\_stack.pop()

**Me falta hacer la parte de read del avance 6**

**Quizá tenga que cambiar los rangos de virtual addresses. Por ahora, hay 333 espacios para cada variable, pero los arrays pueden ocupar hasta 10,000 espacios… Entonces en mis pruebas no puedo crear arrays tan grandes, por precaución.**

La moda no regresa siempre un float, sino que depende del valor de la moda (puede ser int o float en mi caso). Yo asumí que siempre iba a ser float…. Por ahora sólo la casteo a float para evitar problemas