Diseño de compiladores

Proyecto: lenguaje Statlang

Eduardo González Torres

A00826142



17 de noviembre de 2022

Índice

[Descripción del proyecto 2](#_Toc119582685)

[Propósito y alcance del proyecto 2](#_Toc119582686)

[Análisis de requerimientos y descripción de los principales casos de prueba 2](#_Toc119582687)

[Requerimientos funcionales 2](#_Toc119582688)

[Requerimientos no funcionales 3](#_Toc119582689)

[Principales casos de prueba 3](#_Toc119582690)

[Proceso general seguido para el desarrollo 4](#_Toc119582691)

[Bitácora 4](#_Toc119582692)

[Commits 5](#_Toc119582693)

[Reflexión 6](#_Toc119582694)

[Descripción del lenguaje 7](#_Toc119582695)

[Nombre del lenguaje 7](#_Toc119582696)

[Principales características 7](#_Toc119582697)

[Errores que pueden ocurrir 7](#_Toc119582698)

[Compilación 7](#_Toc119582699)

[Ejecución 10](#_Toc119582700)

[Descripción del compilador 10](#_Toc119582701)

[Equipo de cómputo, lenguaje y utilerías usadas 10](#_Toc119582702)

[Descripción del análisis de léxico 10](#_Toc119582703)

[Descripción del análisis de sintaxis 12](#_Toc119582704)

[Descripción de generación de código intermedio y análisis semántico 14](#_Toc119582705)

[Códigos de operación 14](#_Toc119582706)

[Diagramas de sintaxis con puntos neurálgicos y descripción de acciones semánticas y de generación de código 15](#_Toc119582707)

[Tabla de consideraciones semánticas 25](#_Toc119582708)

[Descripción del proceso de administración de memoria en compilación 26](#_Toc119582709)

[Descripción de la máquina virtual 30](#_Toc119582710)

[Equipo de cómputo, lenguaje y utilerías usadas 30](#_Toc119582711)

[Descripción del proceso de administración de memoria en ejecución 30](#_Toc119582712)

[Especificación gráfica y justificación de las estructuras de datos usadas 30](#_Toc119582713)

[Asociación hecha entre las direcciones virtuales y reales 31](#_Toc119582714)

[Pruebas de funcionamiento 31](#_Toc119582715)

Descripción del proyecto

## Propósito y alcance del proyecto

El propósito de este proyecto es desarrollar un compilador y una máquina virtual para un lenguaje que, además de servir como un lenguaje de programación general simple, permita también obtener algunos datos estadísticos básicos de un conjunto de datos numéricos.

## Análisis de requerimientos y descripción de los principales casos de prueba

### Requerimientos funcionales

RF1. El lenguaje soportará operaciones aritméticas básicas (suma, resta, multiplicación y división).

RF2. El lenguaje soportará comparaciones (<, >, diferente que, igual que).

RF3. El lenguaje soportará AND y OR.

RF4. El lenguaje soportará la definición de variables de tipo *int* y *float*.

RF5. El lenguaje soportará el uso de constantes de tipo *int*, *float* y *string*.

RF6. El lenguaje soportará la definición de funciones que regresen valores *int* o *float*.

RF7. El lenguaje soportará cambios de contexto (llamadas a funciones).

RF8. El lenguaje soportará la definición de variables no atómicas de hasta dos dimensiones, cada dimensión con un tamaño máximo de 100.

RF9. El lenguaje soportará el uso de estatutos de asignación, de lectura de archivos JSON y de escritura.

RF10. El lenguaje soportará el uso de reglas de decisión.

RF11. El lenguaje soportará el uso de ciclos.

RF12. El lenguaje soportará el uso de las siguientes funciones estadísticas especiales en variables que sean matrices: obtener la media, obtener la mediana, obtener la moda, obtener la varianza, obtener la desviación estándar, graficar los datos en un histograma, y graficar los datos en un diagrama de caja.

### Requerimientos no funcionales

RNF1. El diseño del lenguaje debe ser tal que los cálculos y accesos a las estructuras de datos se ejecuten eficientemente.

RNF2. El lenguaje incluirá mensajes de error claros si detecta errores de léxico, sintaxis o semántica.

RNF3. El diseño del lenguaje debe ser tal que se minimice el uso de memoria de la máquina virtual, sin sacrificar la legibilidad del código en el que se desarrolle.

### Principales casos de prueba

*Cálculo de factorial (versión cíclica)*

Obtener el factorial de un número dado usando solamente ciclos.

*Cálculo de factorial (versión recursiva)*

Obtener el factorial de un número sin usar ciclos, sólo llamadas a función.

*Serie de Fibonacci (versión cíclica)*

Obtener la serie de Fibonacci hasta cierto número usando solamente ciclos.

*Serie de Fibonacci (versión recursiva)*

Obtener la serie de Fibonacci hasta cierto número sin usar ciclos, sólo llamadas a función.

*Sort para un arreglo*

Ordenar los valores de un arreglo de menor a mayor.

*Find para un arreglo*

Encontrar un valor en un arreglo.

*Multiplicación de matrices*

Obtener el resultado de la multiplicación de dos matrices.

*Conseguir datos estadísticos de un conjunto de datos*

Conseguir la media, mediana, moda, varianza y desviación estándar de un conjunto de datos, así como visualizar el conjunto en un histograma y un diagrama de caja.

## Proceso general seguido para el desarrollo

### Bitácora

*Semana del 26 de septiembre al 3 de octubre*

Trabajé en los avances 0 y 1 del proyecto. Mientras trabajaba en el avance 1 fui notando errores en la especificación del léxico y la sintaxis que había hecho para el avance 0, así que corregí el avance 0 y lo volví a entregar junto con el 1.

*Semana del 4 al 10 de octubre*

Trabajé en los avances 2 y 3 del proyecto. En la entrega que hubo el 10 de octubre sí pude entregar el avance 2 completo.

*Semana del 11 al 17 de octubre*

Trabajé en los avances 3 y 4 del proyecto. Entregué el avance 3 completo el 17 de octubre.

*Semana del 18 al 23 de octubre*

Trabajé en los avances 4 y 5 del proyecto. Completé el avance 4, y del 5 me faltó la parte de la VM.

*Semana del 24 al 29 de octubre*

Debido a que fue semana i, no trabajé en el proyecto.

*Semana del 31 de octubre al 5 de noviembre*

Terminé el avance 5 del proyecto y comencé a trabajar en el 6.

*Semana del 7 al 12 de noviembre*

Terminé el avance 6 y comencé el avance 7, empezando con la parte de ejecución de módulos y arreglos. También realicé las pruebas básicas del proyecto.

*Semana del 14 al 19 de noviembre*

Trabajé en el avance 7 del proyecto: más específicamente, en la generación de código y máquina virtual para las funciones especiales. También arreglé algunos *bugs* e implementé el *read* en la máquina virtual. Comencé la documentación.

*Semana del 20 al 22 de noviembre*

Trabajé en la documentación.

### Commits

git log --pretty=format:"%h - %an, %ar : %s" --reverse

983abba - eduardo139, 6 weeks ago : Commit inicial. Avance 1 completado

8549a78 - eduardo139, 6 weeks ago : Semántica básica de variables

64df681 - eduardo139, 6 weeks ago : Avance 2: cubo semántico, validar que vars existan en VT

bfcf654 - eduardo139, 5 weeks ago : Avance 3 completo

8cc1aee - eduardo139, 4 weeks ago : Work on avance 5 (missing VM part)

4910b72 - eduardo139, 2 weeks ago : Avance 5

1713db3 - eduardo139, 2 weeks ago : Fixed loop quads; work on avance 6

a037438 - eduardo139, 6 days ago : Avance 6: non-atomic types declarations

5b1bca3 - eduardo139, 6 days ago : Avance 6 gen. de código de tipos estructurados

317b948 - eduardo139, 6 days ago : work on arrays in vm (missing verify)

b8f6355 - eduardo139, 5 days ago : Work on avance 7 (up to PARAMETER in vm)

007aae3 - eduardo139, 4 days ago : Work on supporting recursion

852cf7d - eduardo139, 4 days ago : MV ejec de módulos y arreglos

3e2f38d - eduardo139, 3 days ago : Passes most basic tests

7cea0ab - eduardo139, 3 days ago : All basic tests pass

16eb9ee - eduardo139, 3 days ago : Special functions work

c5581b2 - eduardo139, 3 days ago : Expanded maximum array size

51fe82a - eduardo139, 3 days ago : Fixes

c58d9fd - eduardo139, 2 days ago : Added read

2a87a3c - eduardo139, 20 hours ago : Added code documentation

17e5aaa - eduardo139, 23 minutes ago : Fixed error when a loop limit is a float

62ce988 - eduardo139, 63 seconds ago : Added an error message

### Reflexión

Este proyecto fue bastante retador y requirió bastante planeación y esfuerzo. También requirió que entendiera bien los temas que veíamos en clase, y puedo decir que trabajar en el proyecto sí me ayudó bastante a que lo que veíamos en clase me quedara mucho más claro. También termino este proyecto con una mejor comprensión de conceptos básicos que llevo usando desde primer semestre (ifs, ciclos, funciones, etc.), y con nuevos conocimientos sobre las estructuras que están por detrás que ni sabía que existían: directorio de funciones, tablas de variables, cubo semántico, etc…

Creo que en términos de planeación y tiempo este fue el proyecto más demandante de la carrera, así que estoy satisfecho de haber podido cumplir con los entregables y de haber podido completarlo.



# Descripción del lenguaje

## Nombre del lenguaje

El lenguaje se llama Statlang.

## Principales características

Este lenguaje permite declarar variables *int* y *float* de hasta dos dimensiones. Se puede asignar un valor a otro, hacer operaciones aritméticas y de comparación, leer datos de un archivo JSON a una matriz con *read* y mostrar valores a la consola con *write*. También se puede hacer uso de estatutos condicionales con *if* y *else*, y hacerse un ciclo con *from n to m do.* Pueden declararse funciones que reciben cualquier número de parámetros y regresan un valor. Finalmente, el lenguaje cuenta con ciertas funciones especiales para hacer un análisis estadístico básico de una matriz de datos. Se puede obtener la media, la mediana, la moda, la varianza y la desviación estándar de los datos, así como representarlos gráficamente en un histograma o diagrama de caja.

## Errores que pueden ocurrir

### Compilación

*“(variable) does not exist on either the local or global scopes”*

Cuando se intenta usar una variable que no se ha declarado.

*“Cannot assign an element (right operand type) to another that is not of the same type (left operand type)“*

Cuando se intenta asignar (=) un operando de un tipo a algún operando que no es de su mismo tipo.

*“Type mismatch at expression (right operand) (operand) (left operand)”*

Cuando alguna operación no es semánticamente aceptable (se consultó el cubo semántico y se vio que no se acepta).

*“This variable already exists: (variable ID)”*

Cuando se intenta declarar una variable ya declarada.

*“Array size must be between 1 and (maximum dimension size)”*

Cuando se intenta declarar un arreglo con un tamaño de dimensión mayor que el máximo permitido (100).

*“Non-void function (current function name) has no return statement”*

Cuando no se le pone un estatuto *return* a una función que debe tenerlo (porque no es void).

*“This function already exists: (function name)”*

Cuando se intenta declarer una función ya declarada.

*“Non-void function is being called as a statement”*

Cuando se llama una función que regresa un valor, pero no forma parte de ninguna expresión. El valor que regresa se perdería, así que lanzamos un error.

“*Nesting function calls is not supported”*

Debido a que el proyecto no soporta el anidamiento de funciones, se regresa este error si se detecta un intento de anidamiento de funciones.

*“Function (function name) was not declared but is being called”*

Cuando se intenta llamar una función que no fue declarada.

*“The order or types of the parameters in a (function name) call are incorrect”*

Cuando se llama una función con parámetros incorrectos, ya sea debido a que los tipos que se pasan no coinciden con los de la firma, hacen falta parámetros, o hay de más.

*“Return statement in main”*

Cuando se detecta un estatuto *return* en el *main*.

*“Return statement in void function”*

Cuando se detecta un estatuto *return* en una función *void*.

*“Trying to access an index on atomic variable (variable name)”*

Cuando se intenta acceder a una variable atómica como si fuera un arreglo (por índices).

*“Trying to access a 2nd dimension in array (array name)”*

Cuando se intenta acceder a un arreglo como si fuera una matriz (acceder a su segunda dimensión).

*“Tried to access matrix with only one index”*

Cuando se intenta acceder a una matriz como si fuera un arreglo (especificando sólo una dimensión).

“*Void function is being called as part of an expression”*

Cuando se incluye una función *void* en una expresión. Como esta no regresa nada, no debe estar en una expresión.

*“Special functions can only be used on matrices”*

Cuando se intenta usar un *read* o una función especial en algo que no sea una matriz.

### Ejecución

*“Tried to access index (out-of-range index) but index must be between 0 and (dimension size - 1)”*

Cuando se intenta acceder a un índice que está fuera de rango para una variable no atómica.

# Descripción del compilador

## Equipo de cómputo, lenguaje y utilerías usadas

El compilador fue desarrollado en Windows 10, en Python 3, con Visual Studio Code.

Se usaron las librerías *ply* y *json*, *statistics* y *matplotlib.*

## Descripción del análisis de léxico

**program** -> program

**var** -> var

**int** -> int

**float** -> float

**if** -> if

**else** -> else

**from** -> from

**to** -> to

**do** -> do

**mainStart** -> main

**func** -> func

**void** -> void

**read** -> read

**write** -> write

**return** -> return

**mean** -> mean

**median** -> median

**mode**  -> mode

**variance** -> variance

**stddev** -> stddev

**histogram** -> histogram

**boxplot** -> boxplot

**;** -> ;

**,** -> ,

**{** -> \{

**}** -> \}

**(** -> \(

**)** -> \)

**[** -> \[

**]** -> \]

**assignOp** -> =

**lessThanOp** -> <

**greaterThanOp** -> >

**differentOp** -> <>

**equalOp** -> ==

**sumOp** -> \+

**subOp** -> -

**mulOp** -> \\*

**divOp** -> /

**andOp** -> &

**orOp** -> \|

**id** -> [A-Za-z] ([A-Za-z] | [0-9])\*

**cteI** -> [0-9]+

**cteF** -> [0-9]+ \. [0-9]+

**cteString** -> “.+”

## Descripción del análisis de sintaxis

Program -> program id ; ProgramA ProgramB Main

ProgramA -> Vars | ϵ

ProgramB -> Funcion ProgramB | ϵ

Vars -> var VarsA

VarsA -> VarsB ; VarsF

VarsB -> TipoSimple id VarsC

VarsC -> [ cte\_i ] VarsD | ϵ

VarsD -> [ cte\_i ] | ϵ

VarsF -> VarsA | ϵ

Funcion -> func FuncionA id ( FuncionB ) ; FuncionC Bloque

FuncionA -> TipoSimple | void

FuncionB -> Params | ϵ

FuncionC -> Vars | ϵ

Main -> main Bloque

TipoSimple -> int | float

Params -> TipoSimple id ParamsA

ParamsA -> , TipoSimple id ParamsA | ϵ

Bloque -> { BloqueA }

BloqueA -> Estatuto BloqueA | ϵ

Estatuto -> Asignacion | Llamada | Read | Escritura | Condicion | Ciclo | Return | SpecialGraph

Asignacion -> Variable = Exp

Llamada -> id ( Exp LlamadaA )

LlamadaA -> , Exp LlamadaA | ϵ

Read -> read ( cte\_s , id )

Escritura -> write ( EscrituraA EscrituraB )

EscrituraA -> Exp | cte\_s

EscrituraB -> , EscrituraA EscrituraB | ϵ

Condicion -> if ( Exp ) Bloque CondicionA

CondicionA -> else Bloque | ϵ

Ciclo -> from Exp to Exp do Bloque

Variable -> id VariableA

VariableA -> [ Exp ] VariableB | ϵ

VariableB -> [ Exp ] | ϵ

Exp -> Exp1 ExpA

ExpA -> | Exp1 ExpA | ϵ

Exp1 -> Exp2 Exp1A

Exp1A -> & Exp2 Exp1A | ϵ

Exp2 -> Exp3 Exp2A

Exp2A -> Exp2B Exp3 | ϵ

Exp2B -> < | > | <> | ==

Exp3 -> Termino Exp3A

Exp3A -> Exp3B Termino Exp3A | ϵ

Exp3B -> + | -

Termino -> TerminoC TerminoA

TerminoA -> TerminoB TerminoC TerminoA | ϵ

TerminoB -> \* | /

TerminoC -> Factor | Llamada

Factor -> ( Exp ) | cte\_i | cte\_f | Negative\_int | Negative\_float | Variable | Special

Negative\_int -> - cte\_i

Negative\_float -> - cte\_f

Special -> SpecialFunc ( id )

SpecialFunc -> mean | median | mode | variance | stddev

SpecialGraph -> SpecialGraphFunc ( id )

SpecialGraphFunc -> histogram | boxplot

## Descripción de generación de código intermedio y análisis semántico

### Códigos de operación

Los códigos de operación son los siguientes:

|  |  |
| --- | --- |
| Código | Operación |
| 1 | = |
| 2 | < |
| 3 | > |
| 4 | <> |
| 5 | == |
| 6 | + |
| 7 | - |
| 8 | \* |
| 9 | / |
| 10 | & |
| 11 | | |
| 12 | Read |
| 13 | Write |
| 14 | GOTO |
| 15 | GTF |
| 16 | Call (usado para lógica interna) |
| 17 | GOSUB |
| 18 | ERA |
| 19 | PARAMETER |
| 20 | ENDFUNC |
| 21 | Verify |
| 22 | Promedio |
| 23 | Mediana |
| 24 | Moda |
| 25 | Varianza |
| 26 | Desviación estándar |
| 27 | Histograma |
| 28 | Diagrama de caja |

Los rangos de direcciones virtuales son los siguientes:

|  |  |
| --- | --- |
| **Rango** | **Tipo de dato** |
| 100,000 – 199,000 | Int global |
| 200,000 - 299,000 | Float global |
| 400,000 - 499,000 | Int local |
| 500,000 - 599,000 | Float local |
| 700,000 - 799,000 | Int temporal |
| 800,000 - 899,000 | Float temporal |
| 900,000 - 924,999 | Pointer |
| 925,000 – 949,999 | Int constante |
| 950,000 – 974,9999 | Float constante |
| 975,000 – 999,999 | String constante |

### Diagramas de sintaxis con puntos neurálgicos y descripción de acciones semánticas y de generación de código

**Diagram

Description automatically generated**

Program:

1: Se crea el directorio de procedimientos (PD), el diccionario de direcciones virtuales, la tabla de constantes (CT), y el primer cuádruplo (goto main)

2: Se agrega el nombre del programa al PD y se crea su tabla de variables (VT).

**Diagram

Description automatically generated**

Vars:

1: se marca la variable como un array

2: se guarda el tipo

3: se guarda el ID

4: se checa que el tamaño del arreglo esté dentro del rango aceptado. Se añade el valor del tamaño a la tabla de constantes.

5: se checa que el tamaño de la matriz esté dentro del rango aceptado. Se añade el valor del tamaño a la tabla de constantes.

6: Se checa que la variable no exista. Si no existe, se agrega a la VT y se le asigna su dirección virtual (VA).

7: se marca la variable como una matrix

**Diagram

Description automatically generated**

Funcion:

1: se guarda el tipo y se resettean las VAs locales y temporales

2: se checa que la función no exista. Si no existe, se agrega al PD, y se crea en la VT global una variable llamada igual que la función

3: Se guarda en la PD el cuádruplo donde inicia la función

4: Se calculan los requerimientos de memoria de la función y se genera el quad de ENDFUNC

**Diagram

Description automatically generated**

Main:

1: Se resettean las VAs locales y temporales, se rellena el quad 1 (goto main)

2: se calculan los requerimientos de memoria global

**Diagram

Description automatically generatedDiagram

Description automatically generated**

Params:

1: se agrega el tipo de param a la lista de params de la función en el PD

2: se checa que el ID del param no exista en la VT de la función; si no existe, se agrega y se le asigna su VA

**Diagram

Description automatically generated** Diagram

Description automatically generated

Estatuto:

1: Se inicializan las pilas pOper, pilaO y pTypes como vacías.

2: Se verifica que la función llamada sea *void*.

Diagram

Description automatically generated

Asignacion:

1: Se agrega = a pOper.

2: Se genera el cuádruplo correspondiente.

Diagram

Description automatically generated

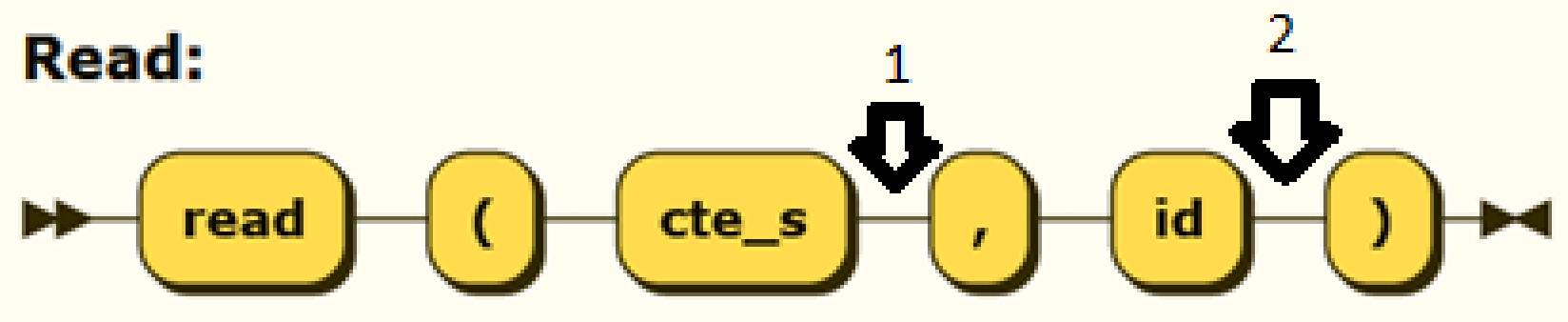
Llamada:

1: Se verifica que la función exista y que la llamada no esté anidada. Se añade ‘call’ a pOper, el ID de la función a pilaO y su tipo a pTypes. Se genera el quad ERA.

2: Se crea una lista de argumentos y otra de tipos de argumentos y se añaden a pilaO y pTypes respectivamente.

3: Se añade el valor de la expresión y su tipo a la lista de argumentos y de tipos de argumentos respectivamente.

4: Se comparan los tipos de los argumentos con la lista de tipos de parámetros de la función que está en el PD. Se genera el cuádruplo correspondiente.

****

Read.

1: Se añade “read” a pOper. Se añade el valor del string y su tipo a pilaO y pTypes respectivamente. Si el string no está en la tabla de constantes, se agrega.

2: Se verifica que la variable exista y que sea una matriz. Se añade a pilaO y pTypes. Se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

Escritura:

1: Se añade “write” a pOper.

2: Se crea una lista de argumentos y otra de tipos de argumentos y se añaden a pilaO y pTypes respectivamente.

3: Se añade el valor de la expresión y su tipo a la lista de argumentos y de tipos de argumentos respectivamente.

4: Se añade el valor del string y su tipo a la lista de argumentos y de tipos de argumentos respectivamente. Si el string no está en la tabla de constantes, se agrega.

5: Se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

Return:

1: Se verifica que no se esté en *main* ni en una función *void*. Se genera el cuádruplo que asigna el valor de la expresión a la variable global de la función.

**Diagram

Description automatically generated**

Condicion:

1: Se genera el cuádruplo de GTF. Se añade el número de cuádruplo a pSaltos.

2: Se rellena el GTF y se genera el GOTO. Se añade el número de cuádruplo a pSaltos.

3: Se rellena el quad correspondiente, ya sea GTF (*if* solo) o GOTO (*if* con *else*).

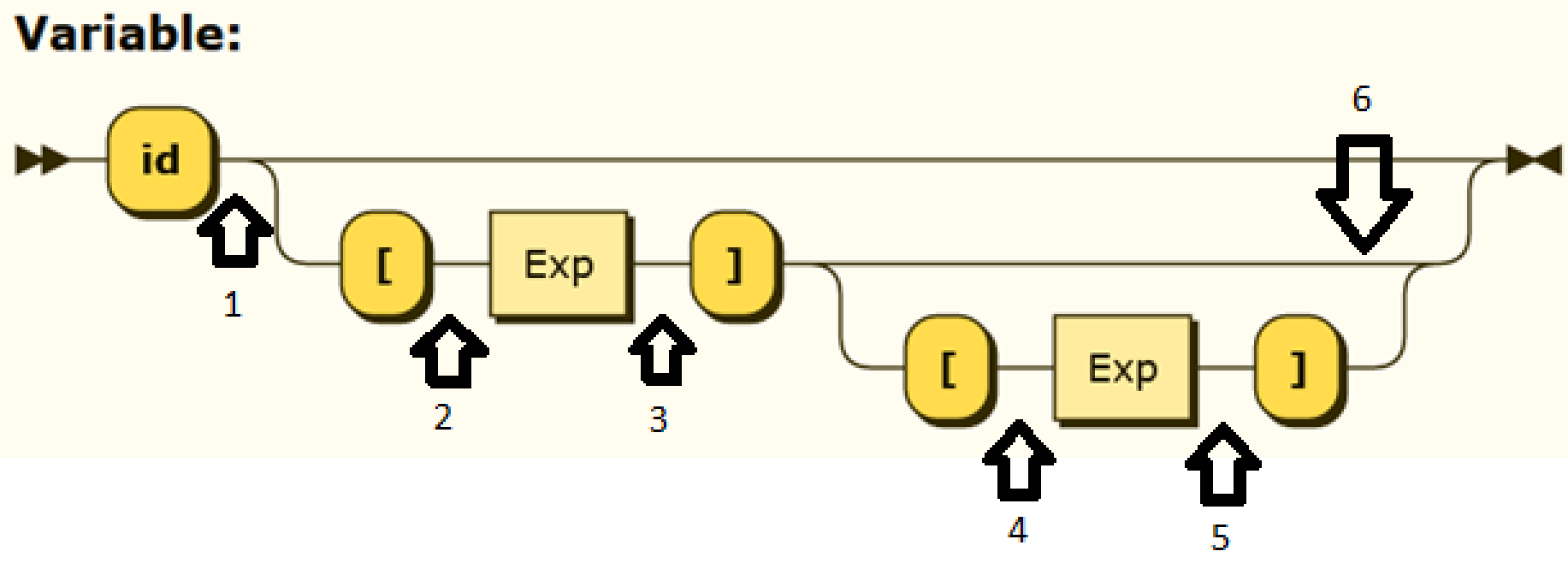
**Diagram

Description automatically generated**

Ciclo:

1: Se genera un quad para restar el lsup del linf. Se agrega 0 a la CT (si no está). Se genera un quad para comparar el resultado de la resta con 0, y otro para un GTF que depende del resultado de la comparación. Se agregan estos dos últimos quads a pSaltos. Se añade la variable contadora a pCiclos.

2: Se obtiene la variable contadora de pCiclos. Se agrega 1 a la CT (si no está). Se suma 1 a la variable contadora (mediante dos quads, uno de + y otro de =). Se genera un GOTO de vuelta al quad de comparación con 0, y se rellena el GTF.

****

Variable:

1: Se checa que la variable exista. Se agrega a pilaO y su tipo a pTypes.

2: Se verifica que la variable sea no atómica. Se agrega a la pilaDim.

3: Se genera el cuádruplo VERIFY. Si es parte de una matriz, se generan los cuádruplos necesarios para hacer el cálculo de dirección. SI no, se generan los cuádruplos de suma para obtener la dirección virtual en forma de pointer.

4: Se verifica que la variable sea una matriz.

5: Se genera el cuádruplo VERIFY para la segunda dimensión, y los cuádruplos de suma para obtener la dirección virtual en forma de pointer.

6: Se verifica que se esté en un arreglo, no una matriz.

**Diagram

Description automatically generated**

Exp:

1: Se añade | a pOper.

2: Si | está en el tope de pOper, se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

Exp1:

1: Se añade & a pOper.

2: Si & está en el tope de pOper, se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

Exp2:

1: Se añade el operador a pOper.

2: Se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

Exp3:

1: Se añade el operador a pOper.

2: Si + o – están en el tope de pOper, se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

Termino:

1: Se verifica que la función llamada no sea *void*.

2: Si \* o / están en el tope de pOper, genera el cuádruplo correspondiente.

3: Se añade el operador a pOper.

Diagram

Description automatically generated

Factor:

1: Se agrega un fondo falso a pOper.

2: Se quita el fondo falso a pOper.

3: Se añade la constante a pilaO y pTypes. Si no está en la CT, se agrega.

Diagram, schematic

Description automatically generated Diagram

Description automatically generatedDiagram

Description automatically generated

Special

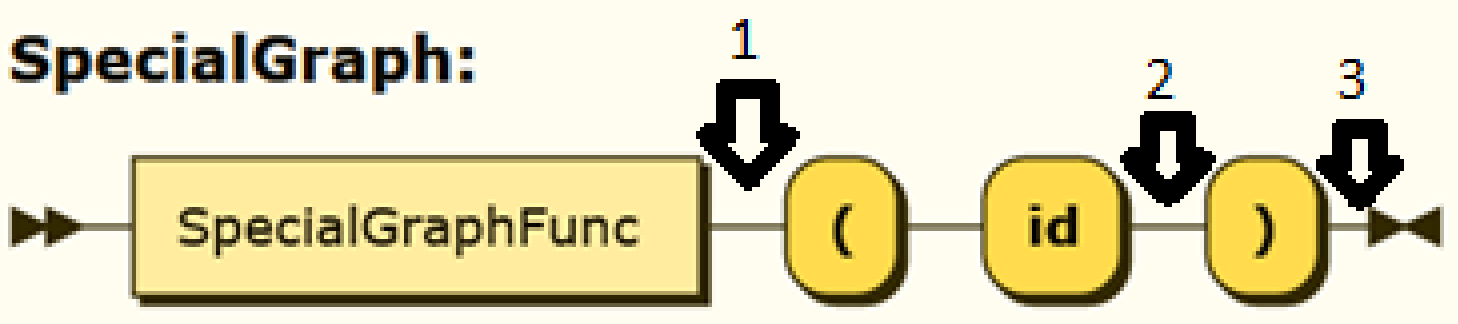
1: Se añade el operador a pOper.

2: Se verifica que la variable exista y que sea una matriz. Se añade a pilaO y pTypes.

3: Se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**



SpecialGraph

1: Se añade el operador a pOper.

2: Se verifica que la variable exista y que sea una matriz. Se añade a pilaO y pTypes.

3: Se genera el cuádruplo correspondiente.

**Diagram

Description automatically generated**

### Tabla de consideraciones semánticas

La tabla de consideraciones semánticas usada es la siguiente:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Left\_op** | **Right\_op** | **\*** | **/** | **+** | **-** | **<** | **>** | **<>** | **==** | **&** | **|** | **=** |
| Int | Int | Int | Float | Int | Int | Int | Int | Int | Int | Int | Int | Int |
| Int | Float | Float | Float | Float | Float | Int | Int | Int | Int | Int | Int | ERR |
| Float | Int | Float | Float | Float | Float | Int | Int | Int | Int | Int | Int | ERR |
| Float | Float | Float | Float | Float | Float | Int | Int | Int | Int | Int | Int | Float |

## Descripción del proceso de administración de memoria en compilación

*Directorio de funciones*

El directorio de funciones se representa con un diccionario. Se eligió un diccionario porque da la capacidad de añadir atributos de todo tipo, y puede accesarse con una llave y no un índice. Tiene la siguiente estructura:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Key** | **Type** | **Params** | **Memory requirements** | **Starting quad.** | **Var. table** |
| Program | Void | [ ] | Dict | None | Dict |
| Function | type | List | Dict | Int | Dict |
| Main | Void | [ ] | { } | Int | Dict |
| … | … | … | … |  | … |

*Tablas de variables*

La tabla de variables se representa con un diccionario. Se eligió un diccionario porque da la capacidad de añadir atributos de todo tipo, y puede accesarse con una llave y no un index. Tiene la siguiente estructura:

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Type** | **Virtual address** | **Dimension** |
| Myvar | Int | Int | None |
| Var2 | Float | Int | None |
| Arr | Int | Int | Dict |
| Mat | Float | Int | Dict |
| … | … | … | … |

Los arreglos tienen, dentro de “Dimension”, otro diccionario donde se guardan los datos relevantes para ellos. Si bien se pudo haber puesto todo en la tabla de variables, se decidió dividirlo de esta manera para que el código fuera más claro. La estructura es la siguiente:

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Lsup** | **m1** | **Next dimension** |
| dim | Int | Int | None o dict |

Next dimension es un diccionario que tienen las matrices, con la siguiente estructura:

|  |  |
| --- | --- |
| **Key** | **Lsup** |
| nextdim | Int |

*Tabla de constantes*

La tabla de constantes se representa con un diccionario, para poder accesar al tipo y VA de una constante con nada más tener su valor. Tiene la siguiente estructura:

|  |  |  |
| --- | --- | --- |
| **Key** | **Type** | **Virtual address** |
| 1 | Int | Int |
| -5.6 | Float | Int |
| “hola” | String | Int |
| … | … | … |

*Cubo semántico*

El cubo semántico se representa con un diccionario. Tiene la siguiente estructura:

|  |  |  |  |
| --- | --- | --- | --- |
| **Key** | **Key2** | **Operador** | **Resultado** |
| “Int” | “Int” | \* | Int |
| / | Float |
| + | Int |
| - | Int |
| < | Int |
| > | Int |
| <> | Int |
| == | Int |
| & | Int |
| | | Int |
| = | Int |
| “Float” | \* | Float |
| / | Float |
| + | Float |
| - | Float |
| < | Int |
| > | Int |
| <> | Int |
| == | Int |
| & | Int |
| | | Int |
| = | ERROR |
| “Float” | “Int” | \* | Float |
| / | Float |
| + | Float |
| - | Float |
| < | Int |
| > | Int |
| <> | Int |
| == | Int |
| & | Int |
| | | Int |
| = | ERROR |
| “Float” | \* | Float |
| / | Float |
| + | Float |
| - | Float |
| < | Int |
| > | Int |
| <> | Int |
| == | Int |
| & | Int |
| | | Int |
| = | float |

*Direcciones virtuales*

Se lleva el control de las direcciones virtuales en un diccionario, para aprovechar el acceso por llave. Tiene la siguiente estructura:

|  |  |  |
| --- | --- | --- |
| **Key** | **Key2** | **Próxima dirección virtual a asignar** |
| “global” | “Int” | Int |
| “Float” | Int |
| “local” | “Int” | Int |
| “Float” | Int |
| “temp” | “Int” | Int |
| “Float” | Int |
| “pointer” | Int |
| “constant” | “Int” | Int |
| “Float” | Int |
| “string” | Int |

*Requerimientos de memoria*

Se representan los requerimientos de memoria de una función con un diccionario que tiene la siguiente estructura:

|  |  |  |
| --- | --- | --- |
| **Key** | **Key2** | **Cantidad de variables de ese tipo** |
| “var” | “Int” | Int |
| “Float” | Int |
| “temp” | “Int” | Int |
| “Float” | Int |
| “pointer” | Int |

*Cuádruplos*

Cada cuádruplo es una lista. Se eligió esta estructura porque un cuádruplo siempre tiene el mismo número y orden de elementos, por lo que no es necesario un diccionario.

[opcode op1 op2 result]

*Lista de cuádruplos*

La lista de cuádruplos que la etapa de compilación genera es una lista de listas. Se usó esta estructura para poder anexar elementos fácilmente, y poder accesarlos por índice.

[ [opcode op1 op2 result], [[opcode op1 op2 result], … ]

*Pilas*

Para realizar la evaluación de expresiones, se usó una pila para operadores, una para operandos, y una para tipos. También se usó la pila de saltos y de dimensiones, como se vio en clase.

# Descripción de la máquina virtual

## Equipo de cómputo, lenguaje y utilerías usadas

La máquina virtual fue desarrollado en Windows 10, en Python 3, con Visual Studio Code.

Se usaron las librerías *json*, *statistics* y *matplotlib.*

## Descripción del proceso de administración de memoria en ejecución

### Especificación gráfica y justificación de las estructuras de datos usadas

Para el manejo de *scope*s, se creó una clase *Memory*. Para instanciar una *Memory* para un contexto global o local, se le pasan al constructor dos parámetros: *type* (“global”, “local” o “constants”) y los requerimientos de memoria de la función.

*Memoria global o local*

Si el atributo *type* es “global” o “local”, entonces el constructor crea los atributos “ints”, “floats”, “temp\_ints”, “temp\_floats”, y “pointers”, que son listas. El constructor pobla cada lista con los registros que necesita el contexto, según viene indicado en los requerimientos de memoria.

*Memoria de constantes*

Si el atributo *type* es “constants”, entonces el constructor sólo crea atributos “ints”, “floats” y “strings”, e inicializa cada uno con una lista vacía. Posteriormente, la máquina virtual instancia una *Memory*, y la pobla con los valores que están en la tabla de constantes.

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Atributos** | **Valor** |
| Global o  local | Ints | List |
| Floats | List |
| Temp\_int | List |
| Temp\_floats | List |
| Pointers | List |
| Constantes | Ints | List |
| Floats | List |
| strings | list |

### Asociación hecha entre las direcciones virtuales y reales

Cuando la máquina virtual (VM) quiere obtener una dirección real a partir de una virtual (VA), hace lo siguiente:

* La VM checa entre qué rangos de direcciones virtuales está la VA dada. En el código, son muchos *if*s del tipo: “si VA > inicio del rango de los ints globales y VA < inicio del rango de los floats globales, entonces…”.
* Una vez que la VM sabe entre qué rango está la VA, regresa la resta de la VA menos el inicio del rango en el que está. Esta es la dirección real.

# Pruebas de funcionamiento

*Cálculo de factorial (versión cíclica)*

program factorial;

var

   int i;

   int j;

   int factorial;

main

{

   i = 1

   j = 1

   factorial = 5

   from 0 to factorial do

   {

        j = j \* i

        i = i + 1

   }

   write("Result is ", j)

}

*Graphical user interface, text

Description automatically generated* *Text

Description automatically generated with low confidence*

*Cálculo de factorial (versión recursiva)*

program factorial;

var

   int factorial;

func int fac (int n);

{

    if (n > 1) {

    return n \* fac(n-1)

    }

    else

    {

    return 1

    }

}

main

{

   factorial = 6

   write("Recursive result is", fac(factorial))

}

*Text

Description automatically generated with medium confidence Text

Description automatically generated with medium confidence*

*Serie de Fibonacci (versión cíclica)*

program fibonacci;

var

   int i;

   int j;

   int n;

   int fibonacci;

main

{

   i = 0

   j = 1

   fibonacci = 9

   write("Loop result is 0, 1...")

   from 2 to fibonacci+1 do

   {

   n=i+j

   i=j

   j=n

   write(n)

   }

}

*Text

Description automatically generated Text

Description automatically generated*

*Serie de Fibonacci (versión recursiva)*

program fibonacci;

var

   int fibonacci;

func int fib (int n);

var

   int m;

   int p;

{

    if (n == 0) {

   return 0

    }

    else {

   if (n == 1) {

            return 1

   }

   else {

       m = fib(n-1)

       p = fib(n-2)

       return m+p

   }

    }

}

main

{

   fibonacci = 9

   write("Recursive result is", fib(fibonacci))

}

A picture containing text

Description automatically generated Graphical user interface, text

Description automatically generated

*Sort para un arreglo*

program vectorsort;

var

   int v[10];

   int i;

   int j;

   int n;

   int t1;

   int t2;

main

{

   write("unsorted")

   i = 0

   from 0 to 10 do

   {

        v[i] = 10-i

   write(v[i])

   i = i+1

   }

   i = 0

   from 0 to 10 do

   {

   j = i+1

        from j to 10 do

   {

       t1 = v[i]

       t2 = v[j]

       if (t1>t2) {

           n = v[i]

      v[i] = v[j]

      v[j] = n

       }

       j = j+1

   }

   i = i+1

   }

   i = 0

   write("sorted")

   from 0 to 10 do

   {

        write(v[i])

   i = i+1

   }

}

0 [14, 1, 'null', 'null']

1 [13, [975000], 'null', 'null']

2 [1, 100010, 925001, 'null']

3 [7, 925001, 925000, 700000]

4 [2, 700000, 925001, 700001]

5 [15, 700001, 18, 'null']

6 [21, 100010, 925000, 700002]

7 [6, 100010, 100000, 900000]

8 [7, 925000, 100010, 700003]

9 [1, 900000, 700003, 'null']

10 [21, 100010, 925000, 700004]

11 [6, 100010, 100000, 900001]

12 [13, [900001], 'null', 'null']

13 [6, 100010, 925002, 700005]

14 [1, 100010, 700005, 'null']

15 [6, 700000, 925002, 700006]

16 [1, 700000, 700006, 'null']

17 [14, 4, 'null', 'null']

18 [1, 100010, 925001, 'null']

19 [7, 925001, 925000, 700007]

20 [2, 700007, 925001, 700008]

21 [15, 700008, 56, 'null']

22 [6, 100010, 925002, 700009]

23 [1, 100011, 700009, 'null']

24 [7, 100011, 925000, 700010]

25 [2, 700010, 925001, 700011]

26 [15, 700011, 51, 'null']

27 [21, 100010, 925000, 700012]

28 [6, 100010, 100000, 900002]

29 [1, 100013, 900002, 'null']

30 [21, 100011, 925000, 700013]

31 [6, 100011, 100000, 900003]

32 [1, 100014, 900003, 'null']

33 [3, 100013, 100014, 700014]

34 [15, 700014, 46, 'null']

35 [21, 100010, 925000, 700015]

36 [6, 100010, 100000, 900004]

37 [1, 100012, 900004, 'null']

38 [21, 100010, 925000, 700016]

39 [6, 100010, 100000, 900005]

40 [21, 100011, 925000, 700017]

42 [1, 900005, 900006, 'null']

43 [21, 100011, 925000, 700018]

44 [6, 100011, 100000, 900007]

45 [1, 900007, 100012, 'null']

46 [6, 100011, 925002, 700019]

47 [1, 100011, 700019, 'null']

48 [6, 700010, 925002, 700020]

49 [1, 700010, 700020, 'null']

50 [14, 25, 'null', 'null']

51 [6, 100010, 925002, 700021]

52 [1, 100010, 700021, 'null']

53 [6, 700007, 925002, 700022]

54 [1, 700007, 700022, 'null']

55 [14, 20, 'null', 'null']

56 [1, 100010, 925001, 'null']

57 [13, [975001], 'null', 'null']

58 [7, 925001, 925000, 700023]

59 [2, 700023, 925001, 700024]

60 [15, 700024, 69, 'null']

61 [21, 100010, 925000, 700025]

62 [6, 100010, 100000, 900008]

63 [13, [900008], 'null', 'null']

64 [6, 100010, 925002, 700026]

65 [1, 100010, 700026, 'null']

66 [6, 700023, 925002, 700027]

67 [1, 700023, 700027, 'null']

68 [14, 59, 'null', 'null']

"unsorted"

10

9

8

7

6

5

4

3

2

1

"sorted"

1

2

3

4

5

6

7

8

9

10

*Find para un arreglo*

program sfm;

var

   int v[10];

   int numToFind;

   int foundNum;

   int i;

   int j;

   int n;

   int t1;

main

{

   write("vector")

   i = 0

   from 0 to 10 do

   {

        v[i] = 10-i

   write(v[i])

   i = i+1

   }

   write("find")

   i = 0

   numToFind = 9

   foundNum = 0

   from 0 to 10 do

   {

   t1 = v[i]

   if (t1 == numToFind) {

       write("found", numToFind)

       write("at index", i)

       foundNum = 1

   }

   i = i+1

   }

   if (foundNum == 0) {

   write("did not find", numToFind)

   }

}

0 [14, 1, 'null', 'null']

1 [13, [975000], 'null', 'null']

2 [1, 100012, 925001, 'null']

3 [7, 925001, 925000, 700000]

4 [2, 700000, 925001, 700001]

5 [15, 700001, 18, 'null']

6 [21, 100012, 925000, 700002]

7 [6, 100012, 100000, 900000]

8 [7, 925000, 100012, 700003]

9 [1, 900000, 700003, 'null']

10 [21, 100012, 925000, 700004]

11 [6, 100012, 100000, 900001]

12 [13, [900001], 'null', 'null']

14 [1, 100012, 700005, 'null']

15 [6, 700000, 925002, 700006]

16 [1, 700000, 700006, 'null']

17 [14, 4, 'null', 'null']

18 [13, [975001], 'null', 'null']

19 [1, 100012, 925001, 'null']

20 [1, 100010, 925003, 'null']

21 [1, 100011, 925001, 'null']

22 [7, 925001, 925000, 700007]

23 [2, 700007, 925001, 700008]

24 [15, 700008, 38, 'null']

25 [21, 100012, 925000, 700009]

26 [6, 100012, 100000, 900002]

27 [1, 100015, 900002, 'null']

28 [5, 100015, 100010, 700010]

29 [15, 700010, 33, 'null']

30 [13, [975002, 100010], 'null', 'null']

31 [13, [975003, 100012], 'null', 'null']

32 [1, 100011, 925002, 'null']

33 [6, 100012, 925002, 700011]

34 [1, 100012, 700011, 'null']

35 [6, 700007, 925002, 700012]

36 [1, 700007, 700012, 'null']

37 [14, 23, 'null', 'null']

38 [5, 100011, 925001, 700013]

39 [15, 700013, 41, 'null']

40 [13, [975004, 100010], 'null', 'null']

"vector"

10

9

8

7

6

5

4

3

2

1

"find"

"found"

9

"at index"

1

*Multiplicación de matrices*

program matrixmult;

var

   int m1[2][2];

   int m2[2][3];

   int mres[2][3];

func void matrixMult();

var

    int i;

    int j;

    int k;

    int m1rows;

    int m1cols;

    int m2cols;

    int m2rows;

    int t1;

    int t2;

    int t3;

{

    i = 0

    m1rows = 2

    m1cols = 2

    m2cols=3

    m2rows=2

    m1[0][0] = 1

    m1[0][1] = -2

    m1[1][0] = 3

    m1[1][1] = 4

    m2[0][0] = 5

    m2[0][1] = -6+3

    m2[0][2] = 7

    m2[1][0] = -8

    m2[1][1] = 9

    m2[1][2] = 10

    from 0 to m1rows do

    {

   j=0

   from 0 to m2cols do

   {

       k=0

       from 0 to m2rows do

       {

      t1 = mres[i][j]

      t2 = m1[i][k]

      t3 = m2[k][j]

      mres[i][j] = t1+t2\*t3

      k = k+1

       }

       j = j+1

   }

   i = i+1

    }

    i = 0

    j = 0

    write("result of matrix multiplication")

    from 0 to m1rows do

    {

   j=0

   from 0 to m2cols do

   {

       write(mres[i][j])

       j = j+1

   }

   i = i+1

    }

}

main

{

   matrixMult()

}

0 [14, 146, 'null', 'null']

1 [1, 400000, 925002, 'null']

2 [1, 400003, 925000, 'null']

3 [1, 400004, 925000, 'null']

4 [1, 400005, 925001, 'null']

5 [1, 400006, 925000, 'null']

6 [21, 925002, 925000, 700000]

7 [8, 925002, 925000, 700001]

8 [21, 925002, 925000, 700002]

9 [6, 925002, 700001, 700003]

10 [6, 700003, 100000, 900000]

11 [1, 900000, 925003, 'null']

12 [21, 925002, 925000, 700004]

13 [8, 925002, 925000, 700005]

14 [21, 925003, 925000, 700006]

15 [6, 925003, 700005, 700007]

16 [6, 700007, 100000, 900001]

17 [1, 900001, 925004, 'null']

18 [21, 925003, 925000, 700008]

19 [8, 925003, 925000, 700009]

20 [21, 925002, 925000, 700010]

21 [6, 925002, 700009, 700011]

22 [6, 700011, 100000, 900002]

23 [1, 900002, 925001, 'null']

24 [21, 925003, 925000, 700012]

25 [8, 925003, 925000, 700013]

26 [21, 925003, 925000, 700014]

27 [6, 925003, 700013, 700015]

28 [6, 700015, 100000, 900003]

29 [1, 900003, 925005, 'null']

30 [21, 925002, 925000, 700016]

31 [8, 925002, 925001, 700017]

32 [21, 925002, 925001, 700018]

33 [6, 925002, 700017, 700019]

34 [6, 700019, 100004, 900004]

35 [1, 900004, 925006, 'null']

36 [21, 925002, 925000, 700020]

37 [8, 925002, 925001, 700021]

38 [21, 925003, 925001, 700022]

39 [6, 925003, 700021, 700023]

40 [6, 700023, 100004, 900005]

41 [6, 925007, 925001, 700024]

42 [1, 900005, 700024, 'null']

43 [21, 925002, 925000, 700025]

44 [8, 925002, 925001, 700026]

45 [21, 925000, 925001, 700027]

46 [6, 925000, 700026, 700028]

47 [6, 700028, 100004, 900006]

48 [1, 900006, 925008, 'null']

49 [21, 925003, 925000, 700029]

50 [8, 925003, 925001, 700030]

51 [21, 925002, 925001, 700031]

52 [6, 925002, 700030, 700032]

53 [6, 700032, 100004, 900007]

54 [1, 900007, 925009, 'null']

55 [21, 925003, 925000, 700033]

56 [8, 925003, 925001, 700034]

57 [21, 925003, 925001, 700035]

58 [6, 925003, 700034, 700036]

59 [6, 700036, 100004, 900008]

60 [1, 900008, 925010, 'null']

61 [21, 925003, 925000, 700037]

62 [8, 925003, 925001, 700038]

63 [21, 925000, 925001, 700039]

64 [6, 925000, 700038, 700040]

65 [6, 700040, 100004, 900009]

66 [1, 900009, 925011, 'null']

67 [7, 925002, 400003, 700041]

68 [2, 700041, 925002, 700042]

69 [15, 700042, 119, 'null']

70 [1, 400001, 925002, 'null']

71 [7, 925002, 400005, 700043]

72 [2, 700043, 925002, 700044]

73 [15, 700044, 114, 'null']

74 [1, 400002, 925002, 'null']

75 [7, 925002, 400006, 700045]

76 [2, 700045, 925002, 700046]

77 [15, 700046, 109, 'null']

78 [21, 400000, 925000, 700047]

79 [8, 400000, 925001, 700048]

80 [21, 400001, 925001, 700049]

81 [6, 400001, 700048, 700050]

82 [6, 700050, 100010, 900010]

83 [1, 400007, 900010, 'null']

84 [21, 400000, 925000, 700051]

85 [8, 400000, 925000, 700052]

86 [21, 400002, 925000, 700053]

87 [6, 400002, 700052, 700054]

88 [6, 700054, 100000, 900011]

89 [1, 400008, 900011, 'null']

90 [21, 400002, 925000, 700055]

91 [8, 400002, 925001, 700056]

92 [21, 400001, 925001, 700057]

93 [6, 400001, 700056, 700058]

94 [6, 700058, 100004, 900012]

95 [1, 400009, 900012, 'null']

96 [21, 400000, 925000, 700059]

97 [8, 400000, 925001, 700060]

98 [21, 400001, 925001, 700061]

99 [6, 400001, 700060, 700062]

100 [6, 700062, 100010, 900013]

101 [8, 400008, 400009, 700063]

102 [6, 400007, 700063, 700064]

103 [1, 900013, 700064, 'null']

104 [6, 400002, 925003, 700065]

105 [1, 400002, 700065, 'null']

106 [6, 700045, 925003, 700066]

107 [1, 700045, 700066, 'null']

108 [14, 76, 'null', 'null']

109 [6, 400001, 925003, 700067]

110 [1, 400001, 700067, 'null']

111 [6, 700043, 925003, 700068]

112 [1, 700043, 700068, 'null']

113 [14, 72, 'null', 'null']

114 [6, 400000, 925003, 700069]

115 [1, 400000, 700069, 'null']

116 [6, 700041, 925003, 700070]

117 [1, 700041, 700070, 'null']

118 [14, 68, 'null', 'null']

119 [1, 400000, 925002, 'null']

121 [13, [975000], 'null', 'null']

122 [7, 925002, 400003, 700071]

123 [2, 700071, 925002, 700072]

124 [15, 700072, 145, 'null']

125 [1, 400001, 925002, 'null']

126 [7, 925002, 400005, 700073]

127 [2, 700073, 925002, 700074]

128 [15, 700074, 140, 'null']

129 [21, 400000, 925000, 700075]

130 [8, 400000, 925001, 700076]

131 [21, 400001, 925001, 700077]

132 [6, 400001, 700076, 700078]

133 [6, 700078, 100010, 900014]

134 [13, [900014], 'null', 'null']

135 [6, 400001, 925003, 700079]

136 [1, 400001, 700079, 'null']

137 [6, 700073, 925003, 700080]

138 [1, 700073, 700080, 'null']

139 [14, 127, 'null', 'null']

140 [6, 400000, 925003, 700081]

141 [1, 400000, 700081, 'null']

142 [6, 700071, 925003, 700082]

143 [1, 700071, 700082, 'null']

144 [14, 123, 'null', 'null']

145 [20, 'null', 'null', 'null']

146 [18, 'matrixMult', 'null', 'null']

147 [17, 'matrixMult', 1, 'null']

"result of matrix multiplication"

21

-21

-13

-17

27

61

*Conseguir datos estadísticos de un conjunto de datos*

program pelos;

var

   float num;

   int i;

   int j;

   float arr[28][30];

   float someNum;

main

{

   i = 0

   j = 0

   num = 6.0

   from 1 to 29 do

   {

   j = 0

        from 1 to 31 do

   {

            arr[i][j] = num

            j = j + 1

       num = num + 20

   }

   i = i + 1

   num = num \* 1.06

   }

   someNum = mean(arr)

   write("mean", someNum)

   someNum = median(arr)

   write("median:", someNum)

   someNum = mode(arr)

   write("mode:", someNum)

   someNum = variance(arr)

   write("variance:", someNum)

   someNum = stddev(arr)

   write("standard deviation:", someNum)

   write("histogram:")

   histogram(arr)

   write("box plot:")

   boxplot(arr)

}

0 [14, 1, 'null', 'null']

1 [1, 100000, 925002, 'null']

2 [1, 100001, 925002, 'null']

3 [1, 200000, 950000, 'null']

4 [7, 925003, 925004, 700000]

5 [2, 700000, 925002, 700001]

6 [15, 700001, 31, 'null']

7 [1, 100001, 925002, 'null']

8 [7, 925003, 925005, 700002]

9 [2, 700002, 925002, 700003]

10 [15, 700003, 24, 'null']

11 [21, 100000, 925000, 700004]

12 [8, 100000, 925001, 700005]

13 [21, 100001, 925001, 700006]

14 [6, 100001, 700005, 700007]

15 [6, 700007, 200001, 900000]

16 [1, 900000, 200000, 'null']

17 [6, 100001, 925003, 700008]

18 [1, 100001, 700008, 'null']

19 [6, 200000, 925006, 800000]

20 [1, 200000, 800000, 'null']

21 [6, 700002, 925003, 700009]

23 [14, 9, 'null', 'null']

24 [6, 100000, 925003, 700010]

25 [1, 100000, 700010, 'null']

26 [8, 200000, 950001, 800001]

27 [1, 200000, 800001, 'null']

28 [6, 700000, 925003, 700011]

29 [1, 700000, 700011, 'null']

30 [14, 5, 'null', 'null']

31 [22, 200001, 840, 800002]

32 [1, 200841, 800002, 'null']

33 [13, [975000, 200841], 'null', 'null']

34 [23, 200001, 840, 800003]

35 [1, 200841, 800003, 'null']

36 [13, [975001, 200841], 'null', 'null']

37 [24, 200001, 840, 800004]

38 [1, 200841, 800004, 'null']

39 [13, [975002, 200841], 'null', 'null']

40 [25, 200001, 840, 800005]

41 [1, 200841, 800005, 'null']

42 [13, [975003, 200841], 'null', 'null']

43 [26, 200001, 840, 800006]

44 [1, 200841, 800006, 'null']

45 [13, [975004, 200841], 'null', 'null']

46 [13, [975005], 'null', 'null']

47 [27, 200001, 840, 'null']

48 [13, [975006], 'null', 'null']

49 [28, 200001, 840, 'null']

"mean"

15647.469708291988

"median:"

12990.492240918278

"mode:"

6.0

"variance:"

143195368.7333757

"standard deviation:"

11966.426732043936

"histogram:"

"box plot:"

**Chart, histogram

Description automatically generated**

**Chart, box and whisker chart

Description automatically generated**