**Instituto Tecnológico y de Estudios Superiores de Monterrey**

Cynthia Castillo A01137090

Sergio Casas A01033864

*Compiladores*

**Documentación Final - Acoustatic**

Dr. José Icaza

Ing. Elda Quiroga

8 de Mayo del 2013

Monterrey Nuevo León, México

**Descripción del Proyecto**

Visión del Proyecto

Acoustatic es un proyecto único e innovador con capacidad de crecer fácilmente. Por su facilidad de uso, flexibilidad y funcionamiento favorece un proceso de aprendizaje continuo y divertido para cualquier usuario. Este utiliza un lenguaje simple y estructurado con un enfoque en el principio de creación de sonidos. Busca aprendizaje en las formas más simples de programación y música aunado así al entretenimiento del usuario.

Objetivo del Proyecto

El objetivo principal es enseñar jóvenes con intereses y principios de computación las bases de la programación. Mediante la utilización de los componentes provistos por el lenguaje, el interesado podrá introducirse poco a poco a la utilización de estructuras de datos manejando por ende un lenguaje de fácil adopción. Dichas estructuras están enfocadas en la implementación de multimedia en las aplicaciones logrando así una introducción a manipulación de otros tipos de archivos que ayudan a desenvolverse en lenguajes más complejos.

Alcance del Proyecto

Nuestro proyecto al estar implementado en fundamentos de Python, nos ayuda a que su sintaxis se pueda ver relacionada con otros lenguajes como son “Patito++” y Java; manteniendo los elementos más simples de los antes mencionados para llevar a cabo su funcionalidad. Estas raíces nos apoyan en la capacidad de creación de arreglos unidimensionales, métodos y ciclos, los cuales enriquecen el funcionamiento y utilidad del lenguaje.

Adicionalmente el lenguaje cuenta con sintaxis especializada para la creación de estructuras ya establecidas las cuales basan su funcionamiento en la creación y modificación de sonidos.

Entre las herramientas adicionales con las cuales cuenta el lenguaje se incluye descripción de errores de sintaxis con aproximación a la línea donde se ejecuta el error y posee conocimiento de las instancias (sean métodos o variables) ya declaradas.

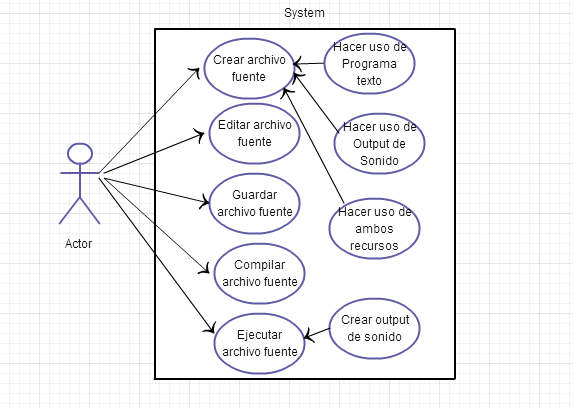
Debido a tales fundamentos, el compilador, provisto con las herramientas y librerías adecuadas es capaz de correr en diferentes tipos de plataformas, como lo son Windows 32 bits y Windows 64 bits.

El lenguaje posee un espacio reducido para la recursión debido a las similitudes que comparte con su lenguaje padre; esta limitante es dependiente en su enteridad a los recursos de la máquina en la que se esté corriendo y desarrollando. Debido a la naturaleza de la creación de multimedia, la interacción que se puede tener con el programa, se ve reducida a sólo la escritura del mismo, dejando de fuera interacción al momento de ejecución. El proyecto no incluye herramientas de asistencia al desarrollo de código de forma dinámica, entregando sólo así material de referencia.

Análisis de Requerimientos

Este lenguaje posee ciertas características y funcionalidades básicas que debe cumplir como lo son estatutos de asignación para el manejo de memoria y variables. Otro requerimiento consiste en la implementación de condiciones y variables condicionales las cuales ayudan a manejar el flujo de datos dentro de la aplicación. Mediante la implementación se es capaz de reducir y modificar rápidamente actividad o problemas que requieran de un esfuerzo considerable. El flujo de datos de input y output se refleja en la lectura del archivo/programa fuente y en la implementación de métodos que asisten en la visualización de información procesada por el proyecto. Las capacidades matemáticas del proyecto se limitan a operaciones básicas de suma, resta, multiplicación, división y remanente. El paso de parámetros y la declaración de procedimientos se atiende mediante la capacidad de manejar variables o constantes dentro de los parámetros a ser usados en la llamada a un procedimiento alterno. Contiene una estructura de memoria la cual permite el cambio de contexto de variables locales manteniendo al mismo tiempo la integridad y valor de las variables globales. El proyecto soporta la capacidad de asignar bloques de memoria de tamaño fijo para la asignación y lectura de espacios individuales del mismo, los cuales pueden ser de diversos tipos de datos, como lo son variables de tipo entero, líneas de texto, entre otros.

Casos de Uso Generales



Test Cases

*Error de Sintaxis*

Algunos de los errores de sintaxis que se crean comúnmente conllevan a la falta de carácter que indica el final de línea y encapsulamiento incompleto.

*Error de falta de declaración de una variable*

Error donde se intenta utilizar una variable que no ha sido declarada y/o inicializada correctamente de manera anterior a la línea donde se ejecuta el error.

*Error de doble declaración de una variable*

En caso contrario, error de tipo local donde una variable intenta ser declarada por más de un elemento y/o procedimiento.

*Realización de procedimientos no acorde con sintaxis*

El intento de llamar a un método que requiere cierto orden y tipo de variable, y al final este recibe variables que no corresponden a las variables pedidas.

*Creación de un archivo fuente con elementos de la línea identificadora faltante*

Falta de elemento de ID, palabra clave “acoustatic” o llave que encapsulan toda la información del proyecto de código.

*Creación de un archivo fuente con falta de una sección “instruments”*

Carencia de la sección obligatoria dentro de instrumentos.

*Creación de una archivo fuente sin declaración de método “play”*

Carencia de un método principal play que busca el uso de las variables instrumento y pieza con el fin de reproducir.

*Creación de un archivo con “play” vacío*

Carencia de una llamada o sector de método el cual pueda ser relacionado con la ejecución de “play”.

*Creación de archivo fuente con carencia de main*

Carencia de una llamada o sector de método el cual pueda ser relacionado con la ejecución del main.

*Declaración de “main” repetidamente*

Creación de un archivo fuente el cual conlleve a la declaración de un procedimiento main en mas de una ocasión.

Proceso General

El proceso de desarrollo consistió principalmente en juntas presenciales de alrededor de 2 horas por sesión, incrementándose conforme se acercaba la fecha de entrega debido a integración de funcionalidad anteriormente no considerada para el proyecto. Nuestro manejo de versiones consistió en el intercambio de archivos.

*Bitácora General*

2013 03 15

Definición de diagramas segunda versión a mano.

Creacion de codigo ejemplo para muestra

2013 03 18

Inicio de definición de tokens y reglas básicas

2013 03 19

Definición de reglas básicas a mano. falta implementarlas en código considerando cambiar de ANTLR a otra cosa

2013 03 20

Inicio de diseño de lista de cuadruplos (pila) y la clase cuádruplo en sí

2013 03 24

Definición de reglas de sintaxis en relación a arreglos y corrección de errores shift reduce

Aprendimos que es necesario hacer reglas vacias para ejecutar código intermedio

2013 03 28

Definición de tipos de variables y sus identificadores

2013 03 29

Definición final de tabla de variables

2013 04 03

Revisión del proyecto con icaza. Vamos atrasados pero le gustó la idea.

2013 04 06

Definición de algunas reglas más de sintaxis,

Corrección de errores conceptuales en el desarrollo de la memoria que me maneja para llevar el conteo de las variables

2013 04 11

Reimplementación de algunos diagramas de railroads, varias reglas han sido modificadas o alteradas

2013 04 21

Implementación de nueva función de error

Lectura del archivo directamente

2013 04 23

Avance con la creación de arreglos. Pareciera que son dependientes de la posición o quien hable después de ellos

2013 04 25

Método de insertar variables creado, este crea los índices para poder llevar el conteo de variables y offsets

Limpieza de código y modularización de ciertas reglas redundantes.

2013 04 26

Funcionando el número de variables por procedimiento, esto es una lista que ayuda a ver que ocupa cada procedimiento y poder hacer comprobación de tipos entre cada una de las variables usadas

Implementado mediante una regla varía al inicio de la regla que dicta a un método

2013 04 27

Implementación de offset adicional para la definición de variables globales debido a un problema para saber si estas corresponden o no a una memoria local. Esto para su uso con métodos

Implementación de reglas de asignación con sus cuádruplos

Creación de reglas para el ciclo del while

2013 04 28

Problemas de implementación de if, por alguna razón no quiere interpretar la regla aun y cuando si está entrando en el if

2013 04 29

Implementación de condicionales if. Esto nos atrasó debido a que no detectaba la regla, resulta que había un espacio de más en la definición del token de if y no agarraba la regla

2013 04 30

Envió de parámetros en sintaxis y creación de las variables correspondientes

Preparación para la implementación del return

Verificación de inicialización de variable

Definición de print en sintaxis para ver ejecución y recolección de datos

2013 05 01

Terminación de sintaxis de arreglos. Se define un tipo que corresponde al arreglo en sí

Verificación de ciclos y creación de su cuádruplo

2013 05 02

Definición de la primera versión del obj de las variables

Implementación de cuádruplos para la asignación de variables

2013 05 03

Definición de formato de memoria que se seguirá para poder enviarlas a la máquina virtual desde el análisis sintáctico

2013 05 04

Implementación de obtención de constantes y obtención de variables desde la máquina sintáctica a la máquina virtual

2013 05 05

Implementación de parámetros y ERA en la máquina virtual,

Finalización de la máquina virtual en su primera modalidad

Implementación de print en VM

2013 05 06

Implementación casi completa de funciones, falta manejo de variables globales y paso de parámetro

2013 05 07

Creación de recursión

Creación de arreglos

Implementación de recursión

Corrección de arreglos

*Reflexión Cynthia*

Definitivamente fue una experiencia un poco agotadora. A final de cuentas el producto final se vió alterado un poco, sin embargo conservó sus bases. Creo que nuestro problema principal fue la falta de información sobre la implementación de código intermedio en la herramienta que utilizamos como parser (PLY). Fuera de eso, siento que aprendí mucho de un lenguaje del que tenía únicamente conocimientos básicos (Python) y que el proyecto se llevó a cabo con éxito.

Reflexión Sergio

Este proyecto me ha dejado una mezcla de sentimientos. Por un lado veo los objetivos y entiendo lo que el curso intenta fomentar. Me agrada mucho como enseña la unión entre distintos niveles de programación y lógica para algo que en el exterior parece muy simple, pero que internamente es el trabajo de años de esfuerzo por parte de una gran cantidad de personas en general.

Sin embargo es un proyecto retador. Un proyecto que te OBLIGA a buscar información referente a tu proyecto desde mucho antes que lo veas en clase. Esto en caso de que efectivamente quieras realizar de manera cómoda y a tiempo las entregas.

En opinión personal, considero que requirió y exigió mucho tiempo extra. Quizá (y lo que es lo más probable) es que no se le dio el suficiente tiempo al inicio del proyecto, entonces se fue acumulando la carga. Otra observación es el hecho de que no es fácil ver la extensión real del proyecto y esto lleva a confiarse.

Por último me gustaría comentar, que veo el objetivo de aprendizaje: no quedarse en el nivel de que los compiladores funcionan con magia, sino comprender el hecho de que internamente su estructura es compleja y requiere de constante mantenimiento. Solo me queda la pequeña observación de la usabilidad de este conocimiento en un ámbito más amplio. Su aplicación es muy específica, y por lo tanto es fácil descuidar este conocimiento adquirido. Espero  poder recordarlo por mucho tiempo

**Descripción del Lenguaje**

Nombre del lenguaje

El nombre del lenguaje es Acoustatic; decidimos que este nombre sería adecuado dadas las acciones que este generaría, sonidos. Acoustatic viene de la palabra acústico (referenciado la producción ,transmisión y recepción de las ondas sonoras) y estático (referenciando el tipo de programación).

Descripción General

Los programas de acoustatic deben comenzar con la palabra “acoustatic” seguido por “piece”, ambas siendo palabras reservadas y a su vez un identificador. Seguido de esto se abrirá un corchete y es aquí donde el usuario puede declarar (o no) variables globales. El programa finaliza cuando se cierra el corchete anterior.

Para la creación de métodos debe de haberse declarado primero una serie de instrumentos. Esta se instancia de la siguiente manera: se utiliza la palabra reservada “new” (que denota que es una nueva serie de instrumentos) seguida por la palabra “instruments” y se abre corchete. Dentro de este va la codificación de sonidos que se especificará más adelante.

Una vez instanciado esto, se procede con la creación de métodos/funciones. Estas pueden ser cero o más y deben declararse antes del “play” y “main”. Las funciones deben comenzar con la palabra reservada “function” seguida por el tipo de variable a retornar (int, fl, string, boolean o bien void si no regresa nada), un id único y paréntesis con parámetros de entrada (no obligatorios). Estos parámetros tienen que tener el tipo de variable y su id seguido por comas (si hay más de uno). Finalmente se agregan las instrucciones a realizar del método y su respectivo return (en caso de tener uno) con un id de una variable ya inicializada o bien una constante.

El método main es muy parecido a las funciones mencionadas anteriormente, con la excepción de que tiene que ir después de “play”, su tipo es void y esta no recibe parámetros de entrada ni tiene valores a regresar. Cabe mencionar que este método es el siguiente en ejecución después de pasar por Play

Las instrucciones a realizar consisten en asignaciones, ciclos, condiciones, llamadas a métodos y modificaciones de sonidos. En las asignaciones tenemos diferentes casos; los más comunes siendo los casos donde se declara un valor alfanumérico a la variable en cuestión. Además de eso es posible definir asignaciones que regresen la ejecución de un método. Por ultimo tenemos la posible asignación de elementos internos de un arreglo a alguna variable alfanumérica o a lo que regrese alguna función en particular, de la misma manera que se hace con variables sencillas.

Posteriormente contamos con los ciclos. En esta ocasión contamos con cyclos while y do while, los cuales, con algunas consideraciones, son capaces de mantener la naturaleza iterativa de un for sin mayor problema. Su sintaxis es simple ya que sólo requiere de la palabra reservada, seguido de una expresión que al final entregue una variable numérica. Adicional, todo el código dependiente de este ciclo debe estar entre llaves ‘{}’.

Seguido existen las condiciones las cuales se encargan de entregar un valor, cierto o falso. Su sintaxis es muy similar a la de los ciclos. Es necesario que venga acompañado de sus llaves. Adicionalmente, puede venir acompañado de un bloque de ejecución en caso de no cumplir con la fase inicial de la condición. De nueva cuenta acompañado de sus llaves

Para las llamadas a funciones existen diversas cosas a considerar. En primera es necesario declarar el tipo de variable que esta regresara. En caso de no regresar ninguna variable, se declara como tipo void y se continua con el bloque.

Este set encapsulado de instrucción, puede ser ejecutado en diversos momentos y desde diferentes lugares del main o de otros métodos. Además pueden proveer una variable de retorno indicada después de la palabra reservada return. Esto puede llegar a ser una variable, una constante o incluso un lugar de un arreglo. Existe la posibilidad también, que llegue a ser una llamada a otra función.

*Descripción de los sonidos*

En relación a sonidos, estos son declarados con 4 atributos distintos. Estos describen la duracion y tipo de sonido que conllevan.   
Posteriormente existen los instrumentos, que son composiciones en arreglos pseudo dinámicos de estos sonidos.

Además de eso, existe una composición más, que consiste en un set de instrumentos. Este es denominado como track y conlleva un set de instrumentos de un mismo tipo de atributo idéntico.

Descripción de los errores

* Variables no definidas
* Variables no inicializadas
* Variables ya instanciadas
* Sonidos no definidos
* Sonidos no inicializados
* Métodos inexistentes
* Parámetros incorrectos (en cuanto a métodos)
* No concordancia en cuanto a tipos

**Descripción del Compilador**

Descripción General

Se utilizó el lenguaje Python para el desarrollo y la herramienta PLY como asistente para el parseo. Para el almacenamiento de variables (tablas de variables) y los directorios de procedimientos se utilizaron diccionarios (ya implementado el lenguaje utilizado). En cuanto a la creación de cuádruplos se utilizaron varias pilas con funciones distintas, las cuales incluyen el almacenamiento de los operadores, operandos, saltos e inclusive para tener un control de los mismos cuádruplos. Se implementó una clase “cuadruplo.py”, la cual almacena el operando, el operador 1, el operador 2 y un resultado. Por último, se agregó la librería sys que permite terminar de manera prematura la ejecución de la compilación y que se utilizó al desplegar los errores que se tenía en el código a compilar.

Descripción de Análisis Léxico

*Palabras reservadas*

'acoustatic':'ACOUSTATIC',

'piece':'PIECE',

'intensity':'INTENSITY',

'pitch':'PITCH',

'duration':'DURATION',

'tone':'TONE',

'new':'NEW',

'Sound':'SOUND',

'sounds':'SOUNDS',

'Instrument':'INSTRUMENT',

'instruments':'INSTRUMENTS',

'create':'CREATE',

'loops':'LOOPS',

'off':'OFF',

'if':'IF',

'for':'FOR',

'else':'ELSE',

'while':'WHILE',

'do':'DO',

'play':'PLAY',

'add':'ADD',

'on':'ON',

'track':'TRACK',

'function':'FUNCTION',

'true':'TRUE',

'false':'FALSE',

'string':'STRING',

'boolean':'BOOLEAN',

'int':'INT',

'fl':'FL',

'print':'PRINT',

'void':'VOID',

'main':'MAIN',

'return':'RETURN'

*Tokens*

OPENCORCH = '\{'

CLOSECORCH = '\}'

OPENBRACK = '\['

CLOSEBRACK = '\]'

OPENPAREN = '\('

CLOSEPAREN = '\)'

COMMA = ','

SEMICOLON = ';'

COLON = ':'

PLUS = '\+'

MINUS = '-'

MULT = '\\*'

DIV = '/'

EE = '=='

EQUALS = '='

REM = '%'

LT = '<'

LE = '<='

GT = '>'

GE = '>='

NE = '<>'

AND = '\&\&'

OR = '\|\|'

NOT = '\!\!'

*Expresiones Regulares*

INTEGER = '\d+'

FLOAT = '((\d\*\.\d+)(E[\+-]?\d+)?|([1-9]\d\*E[\+-]?\d+))'

STRINGLINE = '\".\*?\"'

Descripción de Análisis de Sintáxis

piece: ACOUSTATIC PIECE ID OPENCORCH content CLOSECORCH

content=> declaration content

content=> declaration

declaration=> NEW soundDeclaration

declaration=> NEW instrumentDeclaration

declaration=> NEW trackDeclaration

declaration=> assignment

declaration=> print

print=> PRINT OPENPAREN CLOSEPAREN SEMICOLON

print=> OPENPAREN printContent CLOSEPAREN SEMICOLON

printContent=>casterString STRINGLINE COMMA

printContent=>casterString STRINGLINE

printContent=>idornum COMMA

printContent=>idornum

printContent=>ID OPENBRACK exp2 CLOSEBRACK

soundAttribute=> intensityAttribute SEMICOLON soundAttribute

soundAttribute=> durationAttribute SEMICOLON soundAttribute

soundAttribute=> toneAttribute SEMICOLON soundAttribute

soundAttribute=> pitchAttribute SEMICOLON

soundDeclaration=> SOUND ID OPENCORCH soundAttribute CLOSECORCH

soundDeclaration=> SOUND ID SEMICOLON

intensityAttribute =>INTENSITY COLON exp2

pitchAttribute => PITCH COLON exp2

durationAttribute => DURATION COLON exp2

toneAttribute =>TONE COLON exp2

instrumentDeclaration=> INSTRUMENT ID insertInstrument OPENCORCH soundsPart CLOSECORCH

instrumentDeclaration=> INSTRUMENT ID insertInstrument

soundsPart=>SOUNDS OPENCORCH soundInnerPart CLOSECORCH

soundInnerPart=>ID meteInstr1 COMMA soundInnerPart

soundInnerPart=>NEW soundDeclaration COMMA soundInnerPart

soundInnerPart=>dynamicSoundCreation COMMA soundInnerPart

soundInnerPart=>ID

soundInnerPart=>NEW soundDeclaration

dynamicHelper1=>NEW soundDeclaration

dynamicHelper1=>ID

trackDeclaration=>INSTRUMENTS OPENCORCH trackDeclarationHelper CLOSECORCH playTrack mainMethod

trackDeclaration=>INSTRUMENTS OPENCORCH trackDeclarationHelper CLOSECORCH methodBlock playTrack mainMethod

mainMethod=> FUNCTION VOID MAIN OPENPAREN methodDeclatationParameter CLOSEPAREN OPENCORCH method CLOSECORCH

methodBlock=>methodDeclaration methodBlock

methodBlock=>methodDeclaration

returnStatement=> RETURN idornum SEMICOLON

returnStatement=> RETURN methodCallingParam SEMICOLON

methodDeclaration=> FUNCTION type ID OPENPAREN methodDeclatationParameter CLOSEPAREN OPENCORCH method CLOSECORCH

methodDeclatationParameter=> methodPair COMMA methodDeclatationParameter

methodDeclatationParameter=> methodPair

methodDeclatationParameter=>

methodPair => methodType ID

methodType => SOUND

methodType => BOOLEAN

methodType => INSTRUMENT

methodType => TRACK

methodType => PIECE

methodType => INT

methodType => FL

methodType => STRING

trackDeclarationHelper=>ID COMMA trackDeclarationHelper

trackDeclarationHelper=>NEW instrumentDeclaration COMMA trackDeclarationHelper

trackDeclarationHelper=>ID

trackDeclarationHelper=>NEW instrumentDeclaration

playTrack=>PLAY OPENCORCH playTrackHelper CLOSECORCH

playTrackHelper=>timeSector playTrackHelper

playTrackHelper=> timeSector

playTrackHelper=>methodCalling playTrackHelper

playTrackHelper=>methodCalling

timeSector=>exp2 COLON exp2 OPENCORCH timeSectorHelper CLOSECORCH

timeSectorHelper=>ID SEMICOLON timeSectorHelper

timeSectorHelper=>ID SEMICOLON

timeSectorHelper=>ID COLON LOOPS SEMICOLON timeSectorHelper

timeSectorHelper=>ID COLON LOOPS SEMICOLON

timeSectorHelper=>ID COLON LOOPS OPENCORCH timeSectorHelper2 CLOSECORCH SEMICOLON timeSectorHelper

timeSectorHelper=>ID COLON LOOPS OPENCORCH timeSectorHelper2 CLOSECORCH SEMICOLON

timeSectorHelper2=>ID OFF COMMA timeSectorHelper2

timeSectorHelper2=>ID OFF

methodCalling=>ID verificaMetodo OPENPAREN methodParameter CLOSEPAREN SEMICOLON

methodCalling=>ID verificaMetodo OPENPAREN CLOSEPAREN SEMICOLON

methodCallingParam=>ID verificaMetodo OPENPAREN methodParameter CLOSEPAREN

methodCallingParam=>ID verificaMetodo OPENPAREN CLOSEPAREN

methodParameter=>depthHelperMethod COMMA methodParameter

methodParameter=>depthHelperMethod

depthHelper=>idornum

depthHelper=>idornum COLON depthHelper

depthHelperMethod=>idornumParam

depthHelperMethod=>idornumParam COLON depthHelperMethod

idornumParam=> ID

idornumParam=> INTEGER

idornumParam=> FLOAT

idornumParam=> STRINGLINE

idornumParam=> methodCallingParam

method=>sentence

method=>sentence method

sentence=>soundMod

sentence=>addSound

sentence=>toggleSound

sentence=>condition

sentence=>assignment

sentence=>arrayCreation

sentence=>cycle

sentence=>print

sentence=>methodCalling

sentence=>returnStatement

soundMod=>depthHelper propertyAssignation SEMICOLON

propertyAssignation=>soundProperty COLON exp2

soundProperty=> INTENSITY

soundProperty=> PITCH

soundProperty=>TONE

soundProperty=> DURATION

addSound=>ADD ID COLON ID switch SEMICOLON

addSound=>ADD ID COLON NEW soundDeclaration SEMICOLON

switch=>ON

switch=>OFF

arrayCreation => type ID OPENBRACK num CLOSEBRACK SEMICOLON

type=>SOUND

type=>INSTRUMENT

type=>BOOLEAN

type=>TRACK

type=>PIECE

type=>INT

type=>FL

type=>VOID

type=>STRING

assignment =>simpleAssignment SEMICOLON

assignment=>ID OPENBRACK exp2 CLOSEBRACK EQUALS exp2 SEMICOLON

simpleAssignment=> type ID EQUALS typeAssigmentVerification

simpleAssignment=> ID EQUALS typeAssigmentVerification

typeAssigmentVerification=> booleanExp

typeAssigmentVerification=> casterString STRINGLINE

typeAssigmentVerification=> exp2

typeAssigmentVerification=> methodCallingParam

typeAssigmentVerification=> ID OPENBRACK exp2 CLOSEBRACK

exp2=> t2

exp2=> t2 PLUS exp2

exp2=> t2 MINUS exp2

t2=> idornum

t2=> idornum MULT t2

t2=> idornum DIV t2

t2=> idornum REM t2

t2=> OPENPAREN exp2 CLOSEPAREN

t2=> OPENPAREN exp2 CLOSEPAREN MULT t2

t2=> OPENPAREN exp2 CLOSEPAREN DIV t2

t2=> OPENPAREN exp2 CLOSEPAREN REM t2

num=> INTEGER

num=> FLOAT

cycle=> while

cycle=> doWhile

cycle=> for

while=> WHILE OPENPAREN booleanExp CLOSEPAREN OPENCORCH method CLOSECORCH

doWhile=> DO OPENCORCH method CLOSECORCH WHILE OPENPAREN booleanExp CLOSEPAREN SEMICOLON

booleanExp=> TRUE

booleanExp=> FALSE

booleanExp=> exp2 GT exp2

booleanExp=> exp2 LT exp2

booleanExp=> exp2 GE exp2

booleanExp=> exp2 LE exp2

booleanExp=> exp2 EE exp2

booleanExp=> exp2 NE exp2

booleanExp=> exp2 GT exp2 booleanComparison booleanExp

booleanExp=> exp2 LT exp2 booleanComparison booleanExp

booleanExp=> exp2 GE exp2 booleanComparison booleanExp

booleanExp=> exp2 LE exp2 booleanComparison booleanExp

booleanExp=> exp2 EE exp2 booleanComparison booleanExp

booleanExp=> exp2 NE exp2 booleanComparison booleanExp

booleanComparison =>AND

booleanComparison =>OR

booleanComparison =>NOT

condition=> IF OPENPAREN booleanExp CLOSEPAREN OPENCORCH method CLOSECORCH elseBlock

elseBlock=> ELSE OPENCORCH method CLOSECORCH

elseBlock=>

toggleSound=> depthHelper switch SEMICOLON

idornum=> ID

idornum=> num

Descripción de Generación Código Intermedio y Análisis Semántico

*Código de Operaciones*

1. PR = PilaO.pop(). Crea cuádruplo(29, -1, -1, PR)
2. Agrega variable con su respectivo offset a la Tabla de Variables
3. Agrega método a Direcciones de Procedimientos
4. numCuadr = len(CuadruploPila). Rellena cuádruplo de main (Primer cuádruplo) con el cuádruplo actual
5. Crea cuádruplo (20, -1, -1, “”)
6. Crea cuádruplo (26, -1, -1, -1))
7. Agrega constante a Tabla de Constantes
8. dirMetodoActual = dir(metodoActual ). Crea cuádruplo (24, -1, -1, dirMetodoActual )
9. numCuadruplo = cuad(metodoActual). varGlobal = var(metodoActual). Crea cuádruplo (23, -1, varGlobal, numCuadruplo)
10. PilaO.append(O)
11. arg = PilaO.pop(). Crea cuádruplo(27, ar, -1, numParams)
12. POper.append(O)
13. Linf = 0. Lsup = dimension(ID). tope = PilaO.peek(). Crea cuádruplo(28, tope, Linf, Lsup)
14. aux = PilaO.pop(). T = Temporal. dirBase = dir(PilaDimens.pop()). Inserta la dirBase como constante. base = dir de la constante. Crear cuádruplo(1, aux, base, T). PilaO.append(“(”+T+”)”). quitaFondo = POper.pop()
15. Si POper.peek() == ‘=’. op = POper.pop(). opd2 = PilaO.pop(). opd1. PilaO.pop(). Crea cuádruplo(op, opd2, -1, opd1)
16. SI POper.peek() == ‘\*’ o ‘/’. op = POper.pop(). opd2 = PilaO.pop(). opd1. PilaO.pop(). Crea cuádruplo(op, opd2, -1, opd1)
17. Si POper.peek() == ‘+’ o ‘-’. op = POper.pop(). opd2 = PilaO.pop(). opd1. PilaO.pop(). Crea cuádruplo(op, opd2, -1, opd1)
18. Si POper.peek() == operaciones de comparación. op = POper.pop(). opd2 = PilaO.pop(). opd1. PilaO.pop(). Crea cuádruplo(op, opd2, -1, opd1)
19. Si PilaO.peek() == tipo boolean. res = PilaO.pop(). Crear Cuádruplo (22, res, -1, “”). PSaltos.append(numCuadruplo)
20. Si PilaO.peek() == tipo boolean. res = PilaO.pop(). Crear Cuádruplo (21, res, -1, “”). PSaltos.append(numCuadruplo)
21. fin = PSaltos.pop(). CuadruploPila[fin ].res = numCuad
22. PSaltos.append(numCuad)
23. falso = PSaltos.pop(). ret = PSaltos.pop(). Crea cuádruplo(20, -1, -1, ret). CuadruploPila[fin] = numCuad

Donde sus asosciados eran los siguientes:

1 - +

2 - -

3 - /

4 - \*

5 - <

6 - <=

7 - >

8 - >=

9 - <>

10 - ==

11 - &&

12 - ||

13 - !!

14 - =

15 - %

20 GoTo

21 GoToV

22 GoToF

23 GoSub

24 ERA

25 RETURN

26 RET

27 PARAM

28 VER

29 PRINT

*Diagramas de Sintaxis*

\*Anexos

*Consideraciones Semánticas*

* Las variables tienen que haber sido instanciadas y asignadas primero
* No puedes instanciar dos veces una variable con el mismo ID
* No se utilizan variables fuera del alcance /scope
* Incompatibilidad de tipos
* Los IDs de las variables deben de ser diferentes a los de los métodos

*Cubo/Tabla Semántica*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP1** | **OP2** | **+** | **-** | **\*** | **/** | **%** | **<** | **<=** | **>** | **>=** | **<>** | **!!** | **&&** | **||** | **=** | **==** |
| **Int** | **int** | I | I | I | F | I | B | B | B | B | B | - | - | - | - | B |
| **Int** | **float** | F | F | F | F | F | B | B | B | B | B | - | - | - | - | B |
| **Int** | **bool** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **Int** | **String** | S | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **float** | **int** | F | F | F | F | F | B | B | B | B | B | - | - | - | - | B |
| **float** | **float** | F | F | F | F | F | B | B | B | B | B | - | - | - | - | B |
| **float** | **bool** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **Float** | **String** | S | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **bool** | **int** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **bool** | **float** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **bool** | **bool** | - | - | - | - | - | - | - | - | - | - | - | B | B | - | - |
| **Bool** | **Str** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **Str** | **int** | S | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **Str** | **float** | S | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **Str** | **bool** | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - |
| **Str** | **str** | S | - | - | - | - | - | - | - | - | B | - | - | - | - | B |
| **Int** | **null** | - | - | - | - | - | - | - | - | - | - | - | - | - | I | - |
| **Float** | **null** | - | - | - | - | - | - | - | - | - | - | - | - | - | F | - |
| **Bool** | **null** | - | - | - | - | - | - | - | - | - | - | B | - | - | B | - |

**Descripción de Máquina Virtual**

Descripción General

El equipo de cómputo utilizado para la máquina virtual fue una máquina con procesador AMD6 de 4 núcleos a 64 bits, corriendo sobre windows 7 pro con 4 GB de RAM. El lenguaje utilizado fue Python, en su versión 3.1.1.

También se utilizo la libreria copy contenida en los cores de Python, para la copia de instancias de clases sin modificar el objeto original.

En adición, se importa una conexión con la la librería de sintaxis, la clase cuádruplo, y la clase sys, para poder hacer referencia a la lista de cuadruplos y otros recursos.

Representación Gráfica de Memoria

Clase memoriaAcoustatic

Una clase que en su interior contiene 7 diferentes arreglos. Esta sirve para la descripción de la memoriaLocal y memoriaGlobal

|  |
| --- |
| self.memoriant= [] |
| self.memoriaFloat= [] |
| self.memoriaBoolean= [] |
| self.,memoriaString= [] |
| self.memoriaSonido= [] |
| self.memoriaInstrumento= [] |
| self.memoriaPista = [] |

memoriaConstante y memoriaTemporales

Constantes es un arreglo el cual en su interior tiene muchas tuplas con 3 campos

|  |  |  |  |
| --- | --- | --- | --- |
| offset   |  |  |  | | --- | --- | --- | | tipo (0-7) | valor | offsetContanteAssignado | |
| offset   |  |  |  | | --- | --- | --- | | tipo (0-7) | valor | offsetContanteAssignado | |
| offset   |  |  |  | | --- | --- | --- | | tipo (0-7) | valor | offsetContanteAssignado | |

Al ser trasladado a memoriaConstante pierde su offset, ya que fue asignado en un espacio de memoria real

|  |  |  |
| --- | --- | --- |
| direccionReal   |  |  | | --- | --- | | tipo (0-7) | valor | |
| direccionReal   |  |  | | --- | --- | | tipo (0-7) | valor | |

Algo muy similar sucede con las temporales, sin embargo no existe un proceso de conversión

memoriaHolding

Es una instancia de memoryAcoustatic, la cual sirve para poder meterla en la pila de memorias al momento de hacer cambio de contexto

Params

Params es una lista la cual nos ayuda en el paso de parámetros entre cambio de contextos. Asigna una tupla (tipo valor)

|  |  |
| --- | --- |
| tipo (0-7) | valor |

V

|  |  |
| --- | --- |
| tipo (0-7) | valor |

V

|  |  |
| --- | --- |
| tipo (0-7) | valor |

nextQUad

nextQuad es una pila la cual guarda el siguiente salto, antes de realizar un goSub

|  |
| --- |
| 26 (número de cuádruplo) |
| 26(número de cuádruplo) |
| 24(número de cuádruplo) |

sleepingMemories

SleepingMemories es una pila la cual guarda los contextos antiguos que aún esperan resolución mientras una llamada a método esta siendo ejecutada

|  |  |
| --- | --- |
| |  | | --- | | memoriaMetodoAlfa | |
| |  | | --- | | memoriaMetodoBeta | |

cuadruploArray

Estructura de diccionario (hashing table) en forma de lista que trae consigo representaciones de los cuádruplos. Dentro tiene un arraelo de 4 espacios para cada uno de los elementos

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID(número del cuádruplo) | |  |  |  |  | | --- | --- | --- | --- | | OP | OPD1 | OPD2 | RES | |

Proceso de asociación de memoria

*Inicialización de constantes*

def initializeConstants():

global memoriaConstante

#lectura del archivo que hace el vertedero desde sintaxis

pfile=open("constantes.obj","r")

data = pfile.read()

pfile.close()

lista = data.split("\n")

listaspliteada = []

constantKeys = []

#recoremos para generar los espacios de memoria necesarios

for i in range (0, len(lista)-1):

memoriaConstante.append('')

for item in lista:

#separamos en un arreglo los contenidos de cada linea de la lista

listaspliteada.append(item.split("|"))

for i in range (0, len(listaspliteada)-1):

#quitamos su offset asignado desde la sintaxis y lo guardamos en una direccion real. en base al tipo se le hace el casting necesario antes de instertarlo

pointerConstants = int(listaspliteada[i][2])-14500

if(listaspliteada[i][0] == '0'):

memoriaConstante[pointerConstants] = (0, int(listaspliteada[i][1]))

elif(listaspliteada[i][0] == '1'):

memoriaConstante[pointerConstants] = (1, float(listaspliteada[i][1]))

elif(listaspliteada[i][0] == '2'):

if(listaspliteada[i][1] == 'on' or listaspliteada[i][1] == 'true'):

listaspliteada[i][1] = True

else:

listaspliteada[i][1]= False

memoriaConstante[pointerConstants] =(2, listaspliteada[i][1])

elif(listaspliteada[i][0] == '3'):

memoriaConstante[pointerConstants] =(3, listaspliteada[i][1][1:-1])

else:

print("ERROR DE TIPO")

print(listaspliteada[i])

sys.exit()

*Inicializacion de memogia global*

alocateGlobalVariables()

def alocateGlobalVariables():

#averiguar cuantos espacios vamos a ocupar

#leer e interpretar la primera linea del archivo vpm.obj

punteros = listaspliteada[0]

metodo = int(listaspliteada[0][0])

cantInt =int(listaspliteada[0][1])

cantFloat =int(listaspliteada[0][2])

cantBool =int(listaspliteada[0][3])

cantString =int(listaspliteada[0][4])

cantSonido =int(listaspliteada[0][5])

cantInstrumento =int(listaspliteada[0][6])

cantPista =int(listaspliteada[0][7])

memoriaGlobal.initialize(cantInt, cantFloat, cantBool, cantString, cantSonido, cantInstrumento, cantPista)

#inicializar en cero los espacios correspondientes

Al ser memoria global, es la primera instrucción del archivo y por tanto la 0 en offset. Llama a los métodos propios para la alocación de memoria

*Inicialización de memoria local (en la ejecución de un nuevo método)*

Sigue el mismo principio de alocateGlobalVariables, pero recibe un offset en relación al procedimiento

def startLocalVariables(p):

#tengo que recorrer cada uno de los renglones de listaParametros

#inicializar memoriaLocal del renglon en el cual me encuentro

#averiguar cuantos espacios vamos a ocupar

#leer e interpretar la primera linea del archivo vpm.obj

p = int(p - 13000)

punteros = listaspliteada[p]

[...]

memoriaLocal.initialize(cantInt, cantFloat, cantBool, cantString, cantSonido, cantInstrumento, cantPista)

QuadsAllocation

def quadsAllocation():

global tempQuantity

#print("########################################################")

#abrir el archivo

pfile=open("acoustatic.obj","r")

data = pfile.read()

pfile.close()

lista = data.split('\n')

#recorrer cada linea

for item in lista:

if(item != ""):

if(item[0] == 't'):

tempQuantity = int(item[1:])

else:

#splitear la linea

lineaActual = item.split('|')

#primer elemento es el id

numeroCuadruplo = int(lineaActual[0])

#guardar lo demas en un cuadruplo

cuadruploActual = cuadruplo(lineaActual[1],lineaActual[2],lineaActual[3],lineaActual[4])

#print(numeroCuadruplo)

cuadruploArray[numeroCuadruplo] = cuadruploActual

#print("########################################################")

**Pruebas de Funcionamiento**

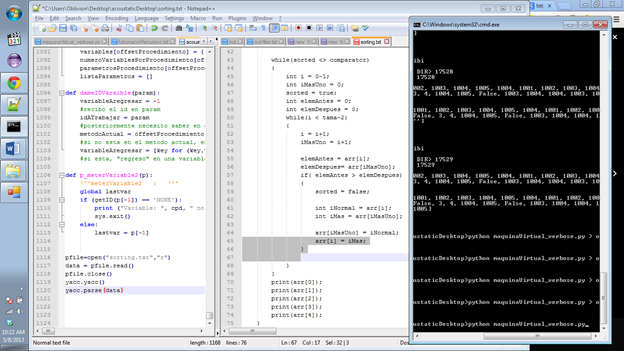
Manejo de Arrays y Sorting

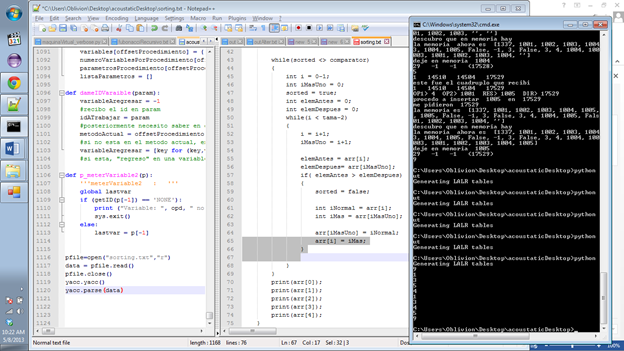
*Codigo*

|  |  |
| --- | --- |
| //Manejo de Arreglos  //Cynthia Castillo  //Sergio Casas  acoustatic piece BusquedaArreglos  {  int nada = 0;  new instruments{  something  }    function int fibonacci(int n){  int x = 0;  }    play{  fibonacci(0);  }  function void main(){  int osom = 1330 + 7;  int arr[5];  arr[0] = 9;  arr[1] = 1;  arr[2] = 3;  arr[3] = 8;  arr[4] = 4;  arr[3] = 5;    boolean sorted = false;  boolean comparator = true;  int tama = 5;    int iterator = 0;  print(arr[0]);  print(arr[1]);  print(arr[2]);  print(arr[3]);  print(arr[4]); | while(sorted <> comparator)  {  int i = 0-1;  int iMasUno = 0;  sorted = true;  int elemAntes = 0;  int elemDespues = 0;  while(i < tama-2)  {  i = i+1;  iMasUno = i+1;    elemAntes = arr[i];  elemDespues= arr[iMasUno];  if( elemAntes > elemDespues)  {  sorted = false;    int iNormal = arr[i];  int iMas = arr[iMasUno];    arr[iMasUno] = iNormal;  arr[i] = iMas;  }    }  }  print(arr[0]);  print(arr[1]);  print(arr[2]);  print(arr[3]);  print(arr[4]);  }  } |

*Codigo Objeto*

|  |  |
| --- | --- |
| t30  0|20|13003|-1|7  1|14|14500|-1|19000  2|14|14500|-1|1001  3|26|-1|-1|-1  4|24|-1|-1|13001  5|27|14500|-1|1  6|23|-1|19001|2  7|1|14501|14502|17500  8|14|17500|-1|1000  9|28|14500|0|4  10|1|14500|14504|17501  11|14|14505|-1|(17501)  12|28|14506|0|4  13|1|14506|14504|17502  14|14|14506|-1|(17502)  15|28|14507|0|4  16|1|14507|14504|17503  17|14|14508|-1|(17503)  18|28|14508|0|4  19|1|14508|14504|17504  20|14|14509|-1|(17504)  21|28|14510|0|4  22|1|14510|14504|17505  23|14|14510|-1|(17505)  24|28|14508|0|4  25|1|14508|14504|17506  26|14|14503|-1|(17506)  27|14|14511|-1|4000  28|14|14512|-1|4001  29|14|14503|-1|1006  30|14|14500|-1|1007  31|28|14500|0|4  32|1|14500|14504|17507  33|29|-1|-1|(17507)  34|28|14506|0|4  35|1|14506|14504|17508  36|29|-1|-1|(17508)  37|28|14507|0|4  38|1|14507|14504|17509  39|29|-1|-1|(17509)  40|28|14508|0|4  41|1|14508|14504|17510  42|29|-1|-1|(17510)  43|28|14510|0|4  44|1|14510|14504|17511  45|29|-1|-1|(17511)  46|9|4000|4001|17512  47|22|17512|-1|84  48|2|14500|14506|17513  49|14|17513|-1|1008 | 50|14|14500|-1|1009  51|14|14512|-1|4000  52|14|14500|-1|1010  53|14|14500|-1|1011  54|2|1006|14507|17514  55|5|1008|17514|17515  56|22|17515|-1|83  57|1|1008|14506|17516  58|14|17516|-1|1008  59|1|1008|14506|17517  60|14|17517|-1|1009  61|28|1008|0|4  62|1|1008|14504|17518  63|14|(17518)|-1|1010  64|28|1009|0|4  65|1|1009|14504|17519  66|14|(17519)|-1|1011  67|7|1010|1011|17520  68|22|17520|-1|82  69|14|14511|-1|4000  70|28|1008|0|4  71|1|1008|14504|17521  72|14|(17521)|-1|1012  73|28|1009|0|4  74|1|1009|14504|17522  75|14|(17522)|-1|1013  76|28|1009|0|4  77|1|1009|14504|17523  78|14|1012|-1|(17523)  79|28|1008|0|4  80|1|1008|14504|17524  81|14|1013|-1|(17524)  82|20|-1|-1|54  83|20|-1|-1|46  84|28|14500|0|4  85|1|14500|14504|17525  86|29|-1|-1|(17525)  87|28|14506|0|4  88|1|14506|14504|17526  89|29|-1|-1|(17526)  90|28|14507|0|4  91|1|14507|14504|17527  92|29|-1|-1|(17527)  93|28|14508|0|4  94|1|14508|14504|17528  95|29|-1|-1|(17528)  96|28|14510|0|4  97|1|14510|14504|17529  98|29|-1|-1|(17529) |





Fibonacci recursivo

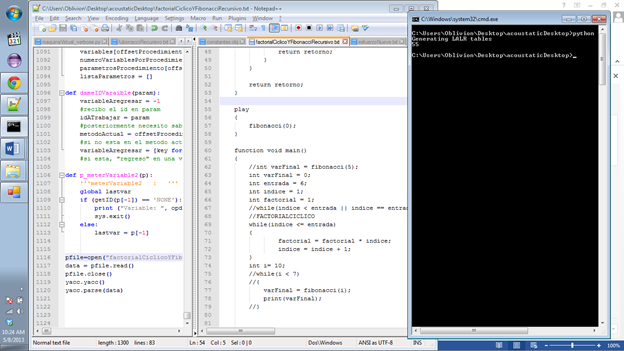
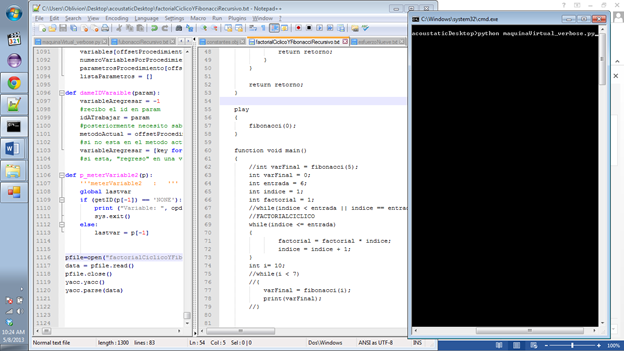
*Codigo:*

|  |  |
| --- | --- |
| //def fib(n):  // if n == 0:  // return 0  //if n == 1:  // return 1  //return fib(n-1) + fib(n-2)  //cynthia castillo  //sergio casas  acoustatic piece esfuerzoNumeroCuatro  {  int nada = 0;  new instruments  {  something  }  function int fibonacci(int n)  {    int retorno = 0;  if(n <= 0)  {  retorno = 0;  //print("REGRESE CERO");  return retorno;  }  else  {  if(n<=1)  {  retorno = 1;  //print("REGRESE UNO");  return retorno;  }  else  {  int menosUno = n-1;  int menosDos = n-2;  //print("numUno");  //print(menosUno);  int num1 = fibonacci(menosUno);  //print("numDos");  //print(menosDos);  int num2 = fibonacci(menosDos);  retorno = num1 + num2;  //print(retorno);  return retorno;  } | }    return retorno;  }    play  {  fibonacci(0);  }  function void main()  {  //int varFinal = fibonacci(5);  int varFinal = 0;  int entrada = 6;  int indice = 1;  int factorial = 1;  //while(indice < entrada || indice == entrada)  //FACTORIALCICLICO  while(indice <= entrada)  {  factorial = factorial \* indice;  indice = indice + 1;  }  int i= 10;  //while(i < 7)  //{  varFinal = fibonacci(i);  print(factorial);  //}      }  } |

*Codigo Objeto*

|  |  |
| --- | --- |
| t8  0|20|13003|-1|33  1|14|14500|-1|19000  2|14|14500|-1|1001  3|6|1000|14500|17500  4|22|17500|-1|8  5|14|14500|-1|1001  6|25|-1|-1|1001  7|20|-1|-1|28  8|6|1000|14501|17501  9|22|17501|-1|13  10|14|14501|-1|1001  11|25|-1|-1|1001  12|20|-1|-1|28  13|2|1000|14501|17502  14|14|17502|-1|1002  15|2|1000|14502|17503  16|14|17503|-1|1003  17|24|-1|-1|13001  18|27|1002|-1|1  19|23|-1|19001|2  20|14|19001|-1|1004  21|24|-1|-1|13001  22|27|1003|-1|1  23|23|-1|19001|2  24|14|19001|-1|1005  25|1|1004|1005|17504 | 26|14|17504|-1|1001  27|25|-1|-1|1001  28|25|-1|-1|1001  29|26|-1|-1|-1  30|24|-1|-1|13001  31|27|14500|-1|1  32|23|-1|19001|2  33|14|14500|-1|1000  34|14|14503|-1|1001  35|14|14501|-1|1002  36|14|14501|-1|1003  37|6|1002|1001|17505  38|22|17505|-1|44  39|4|1003|1002|17506  40|14|17506|-1|1003  41|1|1002|14501|17507  42|14|17507|-1|1002  43|20|-1|-1|37  44|14|14504|-1|1004  45|24|-1|-1|13001  46|27|1004|-1|1  47|23|-1|19001|2  48|14|19001|-1|1000  49|29|-1|-1|1003 |

*Screenshot*



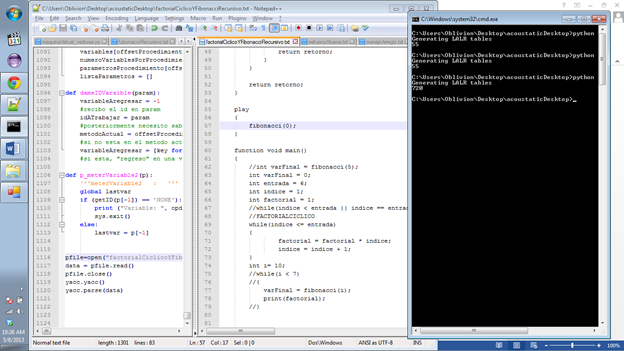
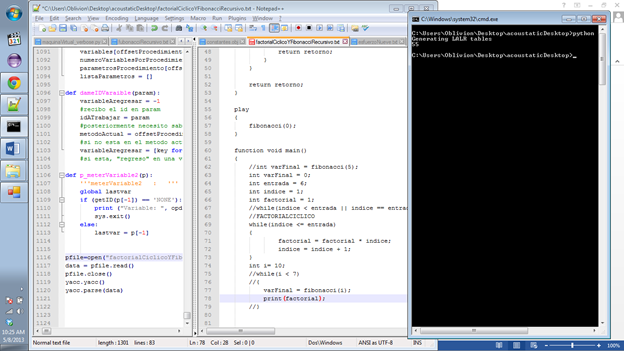
Factorial Ciclico

*Codigo*:

|  |  |
| --- | --- |
| //def fib(n):  // if n == 0:  // return 0  //if n == 1:  // return 1  //return fib(n-1) + fib(n-2)  //cynthia castillo  //sergio casas  acoustatic piece esfuerzoNumeroCuatro  {  int nada = 0;  new instruments  {  something  }  function int fibonacci(int n)  {    int retorno = 0;  if(n <= 0)  {  retorno = 0;  //print("REGRESE CERO");  return retorno;  }  else  {  if(n<=1)  {  retorno = 1;  //print("REGRESE UNO");  return retorno;  }  else  {  int menosUno = n-1;  int menosDos = n-2;  //print("numUno");  //print(menosUno);  int num1 = fibonacci(menosUno);  //print("numDos");  //print(menosDos);  int num2 = fibonacci(menosDos);  retorno = num1 + num2;  //print(retorno);  return retorno;  } | }    return retorno;  }    play  {  fibonacci(0);  }  function void main()  {  //int varFinal = fibonacci(5);  int varFinal = 0;  int entrada = 6;  int indice = 1;  int factorial = 1;  //while(indice < entrada || indice == entrada)  //FACTORIALCICLICO  while(indice <= entrada)  {  factorial = factorial \* indice;  indice = indice + 1;  }  int i= 10;  //while(i < 7)  //{  varFinal = fibonacci(i);  print(factorial);  //}      }  } |

*Codigo Objeto*

|  |  |
| --- | --- |
| t8  0|20|13003|-1|33  1|14|14500|-1|19000  2|14|14500|-1|1001  3|6|1000|14500|17500  4|22|17500|-1|8  5|14|14500|-1|1001  6|25|-1|-1|1001  7|20|-1|-1|28  8|6|1000|14501|17501  9|22|17501|-1|13  10|14|14501|-1|1001  11|25|-1|-1|1001  12|20|-1|-1|28  13|2|1000|14501|17502  14|14|17502|-1|1002  15|2|1000|14502|17503  16|14|17503|-1|1003  17|24|-1|-1|13001  18|27|1002|-1|1  19|23|-1|19001|2  20|14|19001|-1|1004  21|24|-1|-1|13001  22|27|1003|-1|1  23|23|-1|19001|2 | 24|14|19001|-1|1005  25|1|1004|1005|17504  26|14|17504|-1|1001  27|25|-1|-1|1001  28|25|-1|-1|1001  29|26|-1|-1|-1  30|24|-1|-1|13001  31|27|14500|-1|1  32|23|-1|19001|2  33|14|14500|-1|1000  34|14|14503|-1|1001  35|14|14501|-1|1002  36|14|14501|-1|1003  37|6|1002|1001|17505  38|22|17505|-1|44  39|4|1003|1002|17506  40|14|17506|-1|1003  41|1|1002|14501|17507  42|14|17507|-1|1002  43|20|-1|-1|37  44|14|14504|-1|1004  45|24|-1|-1|13001  46|27|1004|-1|1  47|23|-1|19001|2  48|14|19001|-1|1000  49|29|-1|-1|1003 |

*Screenshots*

**Manual de Usuario**

Introducción

El lenguaje de programación Acoustatic es un un lenguaje simple y estructurado con un enfoque en el principio de creación de sonidos. Busca aprendizaje en las formas más simples de programación y música aunado así al entretenimiento del usuario.

Con este lenguaje podrás crear sonidos modificando su tono, volumen, repetición, duración, etc. Así también como modificar la frecuencia de los sonidos y crear conjuntos de instrumentos para formar canciones y piezas.

El primer requerimiento para el funcionamiento adecuado del este lenguajes es la instalación de Python. Una vez instalado esto, se continúa a la ejecución.

La manera en la que ejecuta un programa es escribiendo en la línea de comandos lo siguiente:

*python archivoMaqVir.pyc*

donde Python llama a compilar.

Tipos de Variables

TIPOS DE VARIABLES

int, gloat, string...

Procedimientos

SUMA

RESTA

MULT

PRINT

SOUNDADD

SOUNDMOD

YADAYADA

METODOS

CON/SIN RETURN

CON/SIN PARAMETROS