Trabajo fin de grado sobre la construcción de un entorno de aprendizaje pre-universitario para la programación.

Ezequiel Santamaría Navarro

23 de marzo de 2016

Índice general

Resumen

Proceso de creación de un lenguaje sencillo, de un entorno de programación sin instalación y con herramientas para facilitar el desarrollo.

Capítulo 1

Introducción

1.1. Declaración de intenciones

Mi objetivo sería conseguir que el proyecto cumpla los siguientes puntos:

- El lenguaje debe estar en la medida de lo posible en castellano.
- Debe diferenciar correctamente los distintos tipos de datos.
- No debe asumir automáticamente conversiones de datos.
- Que no requiera instalador.
- Que sea flexible para que los docentes puedan añadir funcionalidad.

Puntos que podrían simplificar el aprendizaje del desarrollo algorítmico y la programación.

1.2. Estado del arte

1.2.1. Descubre

La herramienta Descubre, desarrollada por Juan Antonio Sánchez Laguna, Marcos Menárguez Tortosa, y Juan Antonio Martínez Navarro.
La herramienta se compone de:

- Un editor de texto, derivado de CodeMirror.
- Un lenguaje de programación propio, derivado de Java, con un las funcionalidades de processing.
- Un entorno con un lienzo para dibujar y una salida para escribir texto.
- Un repositorio de usuarios y programas que permite copiar el código de otros para hacer clones.

Con los siguientes puntos positivos:

- Entorno completo. Desde escribir el código hasta ver su resultado en la misma página.
- Sin instalación y sin necesidad de conocimientos sobre compilaciones.
- El lenguaje comparte sintaxis con C y Java, lo que es útil para avanzar más allá de la enseñanza.
- La página tiene una lista de tutoriales y documentación sobre la funcionalidad de processing.

Y los siguientes puntos negativos:

- El lienzo es un poco pequeño. Se limita a 320x320.
- La sintaxis de iJava no es amigable para nuevos alumnos.
- La ejecución del código es síncrono. Esto reduce las posibilidades tecnológicas. Como por ejemplo, no se pueden cargar imágenes en tiempo de ejecución.
- No se puede extender el lenguaje con el propio lenguaje. Es decir, no se puede escribir un módulo para que se incluya en otro código. Esta parte no es importante para el alumno, pero sí para el docente que quiera extender el lenguaje.

Acceso: `http://descubre.inf.um.es`

1.2.2. Codecombat

CodeCombat es un proyecto de código abierto para aprender a programar en base a un juego multijugador. Actualmente tiene varios problemas sencillos (que van creciendo en dificultad) para ir aprendiendo los mecanismos de los distintos lenguajes que ofrece, a la vez que vas ganando monedas para poder comprar objetos que ofrecen mayor funcionalidad (te permiten usar más métodos para resolver los distintos problemas).

Se pueden destacar los siguientes puntos del proyecto:

- Gran conjunto de problemas a resolver que van creciendo en dificultad.
- Permite seleccionar entre Python, JavaScript, Clojure, CoffeScript y Lua.
- Acabado artístico profesional, así como una gran cantidad de objetos.
- Proyecto de código abierto.
- Traducido (parcialmente) a varios idiomas, entre los cuales se encuentra el español.

Sin embargo, el objetivo de esta plataforma es aprender resolviendo problemas concretos como moverse por un laberinto, o eliminando unos enemigos, pero no crear otras aplicaciones. Aún así, es una muy buena herramienta de aprendizaje.

Acceso: `http://codecombat.com`

1.2.3. Scratch

1.2.4. code.org

Capítulo 2

Lenguaje zl

2.1. Introducción

El lenguaje está enfocado a cumplir las siguientes propiedades:

- Enfocado a parecerse sintácticamente al pseudocódigo en castellano, parecido al enseñado en la asignatura IP.
- Diferenciando los tipos de datos, sin hacer implícitas sus conversiones ni sus operaciones.
- Insensible a mayúsculas y minúsculas, y aceptando tildes y eñes en los nombres.
- Con un único tipo de datos para representar números enteros y decimales (similar a Javascript).

2.2. Definición formal del lenguaje

La gramática del lenguaje en formato BNF es:

```
; Insensible a mayúsculas y minúsculas
; Notación BNF
; <_> significa espacio en blanco (cualquier tipo de espacio en blanco), o un comentario
; Algunos de los espacios son opcionales, allá donde no haya ambiguedad:
; por ejemplo: a.b o a<-b


; Reglas gramaticales
; nombreSimple expresión regular ([A-Za-záéíóúÁÉÍÓÚñÑ][A-Za-záéíóúÁÉÍÓÚñÑ0-9]*)
; numero expresión regular ((?:[0-1]+(?:\|2))|(?:[0-9A-Fa-f]+(?:\|16))|(?:[0-9]+(?:\|10)?))
; texto expresión regular \"([^"\\]|\\.|\\\n)*\"
; letra expresión regular \'([^"\\]|\\.|\\\n)*\'
```

```
; comentario expresión regular \/\/.*

; Nota, para las expresiones del mismo nivel,
; el arbol que se construye se reordena para evaluar primero a la izquierda.

<modulo> ::= [<configuraciones> <_>] <subrutina>+

<configuraciones> ::= "configuracion" <_> (<configuracion> <_>)* "fin"

<configuracion> ::= "importar" <_> <texto>
    | "integrar" <_> <texto>
    | <nombre> <_> "<-" <_> <numero>
    | <nombre> <_> "<-" <_> <texto>

<subrutina>  ::= <subrutinaCabecera> <_> <subrutinaCuerpo>

<subrutinaCabecera>  ::= "subrutina" (<_> <modificador>)* [<_> <nombre>]

<subrutinaCuerpo>  ::= <datos> <_> <algoritmo> <_> "fin"

<modificador> ::= "interna"
    | "primitiva"
    | "conversora"

<nombre> ::= <nombreSimple> (<_> "." <_> <nombre>)*

<datos> ::= "datos" <_> (<declaracion> <_>)+

<algoritmo> ::= "algoritmo" <_> (<sentencia> <_>)+

<declaracion> ::= <nombre> <_> "es" <_> <nombre> (<_> <declaracionModificador>)*
    | <nombre> <_> "es" <_> "lista" "(" <_> <listaDecl> <_> ")" <_> "de" <_> <nombre> (<_> <declaracionModificado
    | <nombre> <_> "es" <_> "relacion" <_> "de" <_> <nombre> <_> "a" <_>  (<_> <declaracionModificador>)*

<listaDecl> ::= <rango> <_> "," <_> <listaDecl>
    | <rango>

<rango> ::= <numero> <_> ".." <_> <numero>

<sentencia> ::= <asignacion>
    | <llamada>
    | <repetir>
    | <sicondicional>
    | <mientras>
    | "pausar"

<declaracionModificador> ::= "de" <_> "entrada"
    | "de" <_> "salida"
    | "global"

<asignacion> ::= <nombre> <_> "<-" <_> <expresion>
    | <nombre> <_> "(" <_> <listaAcceso> <_> ")" <_> "<-" <_> <expresion>

<llamada> ::= <nombre> <_> "->" <_> <nombre> <_> "->" <_> <nombre>
    | <nombre> <_> "->" <_> <nombre>
    | <nombre> <_> "[" <_> (<llamadaAsignacion> <_>)+ "]"
```

```
<repetir> ::= "repetir" <_> <expresion> <_> "veces" <_> (<sentencia> <_>)+ "fin"

<mientras> ::= "mientras" <_> <expresion> <_> "hacer" (<sentencia> <_>)+ "fin"

<sicondicional> ::= "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ "fin"
    | "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sinocondicional>
    | "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sino>

<sinocondicional> ::= "o" <_> "si" <_> <expresion> <_> "hacer" <_> (<sentencia> <_>)+ "fin"
    | "o" <_> "si" <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sinocondicional>
    | "o" <_> "si" <expresion> <_> "hacer" <_> (<sentencia> <_>)+ <sino>

<sino> ::= "si" <_> "no" <_> "hacer" <_> (<sentencia> <_>)+ "fin"

<llamadaAsignacion> ::= <expresion> <_> "->" <_> <nombre>
    | <nombre> <_> "<-" <_> <expresion>

<expresion> ::= <expresionTercera> <_> <operadorBinarioCuarto> <_> <expresion>
    | <expresionTercera>

<expresionTercera> ::= <expresionSegunda> <_> <operadorBinarioTercero> <_> <expresionTercera>
    | <expresionSegunda>

<expresionSegunda> ::= <expresionPrimera> <_> <operadorBinarioSegundo> <_> <expresionSegunda>
    | <expresionPrimera>

<expresionPrimera> ::= <operadorUnario> <_> <expresion>
    | <evaluacion> <_> <operadorBinarioPrimero> <_> <expresionPrimera>
    | <evaluacion>

<evaluacion> ::= <numero>
    | <texto>
    | <letra>
    | "verdadero"
    | "falso"
    | <evaluacion> <_> "como" <_> <nombre>
    | <nombre> <_> "(" <_> <listaAcceso> <_> ")"
    | <nombre>
    | "(" <_> <expresion> <_> ")"

<listaAcceso> ::= <rango> <_> "," <_> <listaAcceso>
    | <expresion> <_> "," <_> <listaAcceso>
    | <rango>
    | <expresion>

<operadorUnario> ::= "-"
    | "+"
    | "no"

<operadorBinarioCuarto> ::= "y"
    | "o"

<operadorBinarioTercero> ::= "<"
    | ">"
    | "<="
    | ">="
    | "="
```

```
    | "<>"

<operadorBinarioSegundo> ::= "+"
    | "-"

<operadorBinarioPrimero> ::= "*"
    | "/"
    | "%"
```

Cabe destacar de la gramática las 4 expresiones y los cuatro grupos de operadores, para poder definir correctamente el orden en el cuál los operadores se reducen.

Por otro lado, los número en el lenguaje permiten indicar base 2, 10 o 16 de la siguiente manera:

- 51966 como número en base 10
- 51966|10 como el mismo número
- $CAFE$|16 o $cafe$|16 como el mismo número en base hexadecimal.
- 1100101011111110|2 como el número en base binaria.