

Lenguajes de Programación
Librillo de Reportes de Proyectos.

José Velez
Carlos Ramirez
Keyla Figueroa

3 de septiembre de 2013

Índice general

1. Prefacio	2
2. Introducción	3
3. Proyecto 1	4
3.1. Requerimientos del Proyecto Qt Sudoku:	5
3.2. Herramientas Utilizadas en el Desarrollo del Proyecto:	6
3.3. GitHub:	7
3.4. Doxygen:	8
3.5. Objetivos:	9
3.6. Desarrollo:	9
3.7. Conclusiones:	10
3.8. Referencias:	10
4. Proyecto 2	11
4.1. Requerimientos del Proyecto PySudoku:	12
4.2. Herramientas Utilizadas en el Desarrollo del Proyecto:	13
4.3. GitHub:	14
4.4. Doxygen:	15
4.5. Objetivos:	16
4.6. Desarrollo:	16
4.7. Conclusiones:	17
4.8. Referencias:	17
5. Proyecto 3	18
5.1. Requerimientos del Proyecto Analizador Léxico:	19
5.2. Herramientas Utilizadas en el Desarrollo del Proyecto:	20
5.3. GitHub:	21
5.4. Objetivos:	22
5.5. Desarrollo:	22
5.6. Conclusiones:	23

<i>ÍNDICE GENERAL</i>	2
5.7. Referencias:	23
6. Glosario	24

Capítulo 1

Prefacio

Este librito pretende mostrar los objetivos, los problemas presentados en el transcurso del curso de Lenguajes de Programación, en el desarrollo de los distintos proyectos planteados, mostrar las ventajas y desventajas de las distintas herramientas empleadas para el desarrollo de las aplicaciones propuestas, las particularidades y como se resolvía los problemas presentados en la implementación de cada aplicación.

Por otro lado, también brinda un resumen de las experiencias en cada herramienta aprendida a lo largo de Curso de Lenguajes de Programación.

Con el fin de aprovechar al máximo la información de este documento, se explica y se detalla la utilidad de cada herramienta utilizadas tanto para programar, el control de versionamiento y documentación de las aplicaciones desarrolladas.

Capítulo 2

Introducción

El Curso de Lenguajes de Programación dictado en Término I -2013 por el Ing. Xavier Tibau, pretende obtener como resultados, conocer la sintaxis y semántica de los lenguajes de programación. Entender el rol del hardware en la implementación semántica del lenguaje. Comprender la gramática de los lenguajes para poder comparar sus características y escoger el lenguaje más apropiado para una aplicación. Escribir programas utilizando cada lenguaje de diferentes paradigmas que permitan demostrar sus características y similitudes.

Capítulo 3

Proyecto 1

El proyecto de Qt sudoku fue desarrollado en C++ QT.



Figura 3.1: qt

El objetivo del primer Proyecto del Curso de Lenguajes de Programación fue desarrollar un Sudoku, en QT como herramienta de desarrollo junto a Git como herramienta de versionamiento y el uso de doxygen como herramienta de documentación.

3.1. Requerimientos del Proyecto Qt Sudoku:

Implementar una aplicación de Sudoku en Qt, en la cual podamos validar si la resolución de un tablero es válida o no es válida.

Nuestra aplicación debe generar tableros de sudoku con distintas dificultades.

Nuestra aplicación debe mostrarnos ayuda durante la resolución del tablero mostrándonos, cuando un número ingresado es incorrecto si ya existe en la misma fila, columna o cuadrante

Nuestra aplicación debe ofrecernos la opción de mostrar pistas, la cual nos indicará si los numeros ingresados son correctos o incorrectos validando el tablero.

Nuestra aplicación nos permitirá llevar un registro de los mejores puntajes obtenidos al resolver el tablero en sus diferentes dificultades.

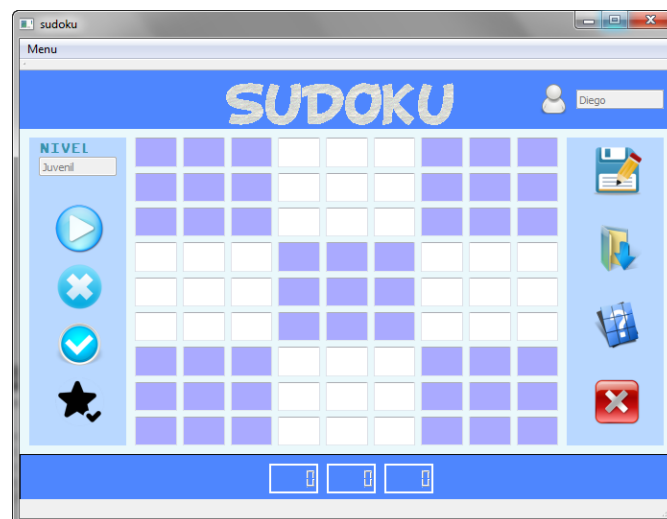


Figura 3.2: Tablero Sudoku

3.2. Herramientas Utilizadas en el Desarrollo del Proyecto:

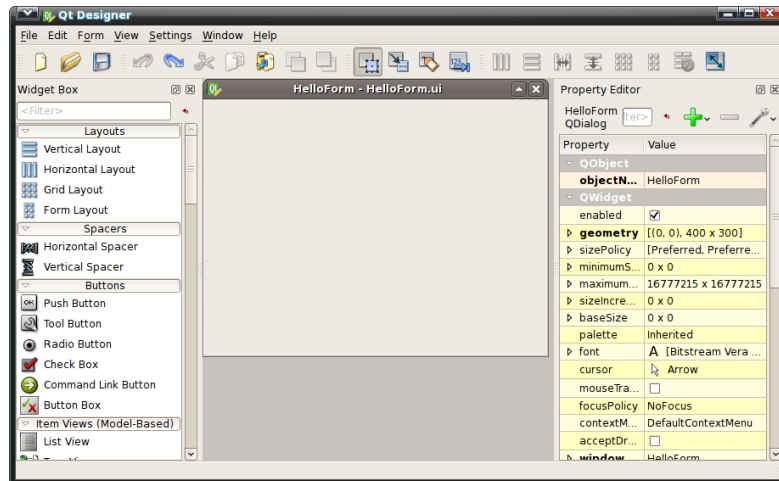


Figura 3.3: qt

Qt es una biblioteca multiplataforma ampliamente usada para desarrollar aplicaciones con interfaz gráfica de usuario, así como también para el desarrollo de programas sin interfaz gráfica, como herramientas para la línea de comandos y consolas para servidores. Qt es desarrollada como un software libre y de código abierto a través de Qt Project, donde participa tanto la comunidad, como desarrolladores de Nokia, Digia y otras empresas. Anteriormente, era desarrollado por la división de software de Qt de Nokia, que entro en vigor después de la adquisición por parte de Nokia de la empresa noruega Trolltech, el productor original de Qt, el 17 de junio de 2008.³ Qt es distribuida bajo los terminos de GNU Lesser General Public License (y otras). Por otro lado, Digia está a cargo de las licencias comerciales de Qt desde marzo de 2011.

3.3. GitHub:



Figura 3.4: git

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código se almacena de forma publica, aunque también se puede hacer de forma privada, creando una cuenta de pago.

3.4. Doxygen:

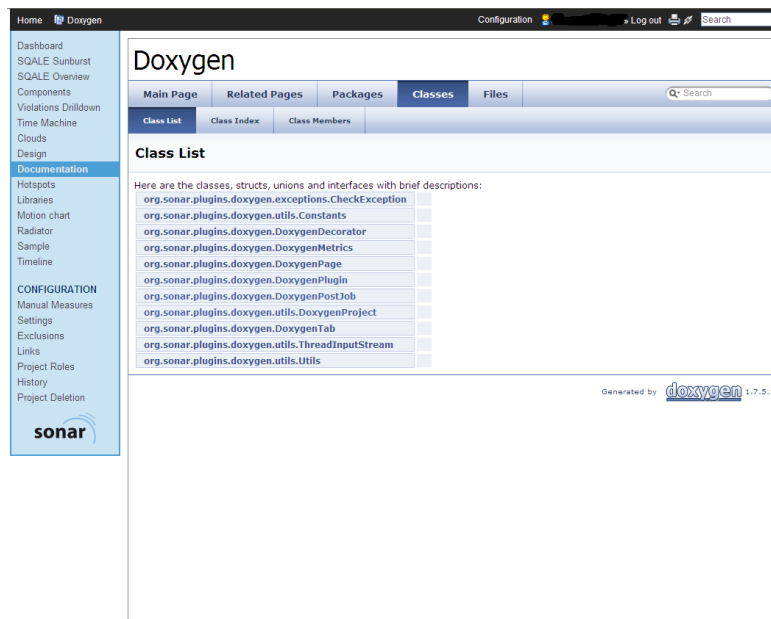


Figura 3.5: doxygen

Doxygen es un generador de documentación para C++, C, Java, Objective-C, Python, IDL (versiones Corba y Microsoft), VHDL y en cierta medida para PHP y D. Dado que es fácilmente adaptable, funciona en la mayoría de sistemas Unix así como en Windows y Mac OS X. La mayor parte del código de Doxygen está escrita por Dimitri van Heesch. Doxygen es un acrónimo de dox(document) gen(generator), generador de documentación para código fuente. Varios proyectos como KDE usan Doxygen para generar la documentación de su API. KDevelop incluye soporte para Doxygen.

3.5. Objetivos:

Crear una aplicación de sudoku.

Utilizar QT como principal herramienta de desarrollo.

Entender el funcionamiento de los signals y slots en QT.

Implementar la aplicación con la metodología Orientada a objetos.

Utilizar GIT como herramienta de control de versionamiento.

Utilizar Doxygen para documentacion del código.

3.6. Desarrollo:

El desarrollo de este primer proyecto se realizó entre los tres integrantes del grupo, empezando por crear la interfaz de la aplicación, como primer paso crear un tablero implementado con una matriz de QTextEdit .

El trabajo fue dividido entre los tres integrantes del grupo, Diego Cabrera desarrollo la interfaz y la documentación de la aplicación.

La funcionalidad fue dividida entre Carlos Ramírez y Joselez.

José Vélez desarrollo el iniciar partida el cual generaba tableros a partir de una plantilla otra de las funcionalidades implementadas fue el cargar y guardar partida, debidamente encriptado con operaciones de desplazamiento al código ASCII de los valores del tablero de sudoku. También se encargo de implementar los puntajes que se calculaba en base al tiempo tomado en resolver el juego.

Carlos Ramírez se encargo de implementar la validación del tablero lleno si era correcto o incorrecto, también de la parte de corrección in game de los números ingresados en el tablero por medio del evento textChanged y de la opción de dar pista que permitía mostrar al usuario que valor ingresado era correcto y cual no lo era antes de terminar de llenar el tablero.

3.7. Conclusiones:

La experiencia en QT, como herramienta de trabajo fue satisfactoria debido a que se orientaba a objetos, y su fácil entendimiento por la organización al crear las clases y manejar la funcionalidad por medio de los signals y slots, que nos permit conectar un evento o cambio de estado en un objeto con una determinada función de la aplicación. El uso de la herramienta GIT es muy importante en el desarrollo de aplicaciones en grupo, ya que nos permite tener una referencia de la colaboración de cada integrante, y una manera organizada de realizar cambios a nuestra aplicacievando un registro ordenado de las versiones realizadas a lo largo del proceso de desarrollo.

3.8. Referencias:

<http://www.zonaqt.com>

<http://git-scm.com/book/es>

<http://www.stack.nl/~dimitri/doxygen/manual/>

Capítulo 4

Proyecto 2

El proyecto de Pysudoku fue desarrollado en Python.



Figura 4.1: qt

El objetivo del segundo Proyecto del Curso de Lenguajes de Programación fue cambiar de lenguaje nuestro sudoku en QT a Python, en Python que se encontro cambio en la sintaxis instrucciones, un lenguaje mas estructurado por identaciones y una particularidad en el uso de variables se continuó trabajando junto a Git como herramienta de versionamiento y el uso de doxygen como herramienta de documentación.

4.1. Requerimientos del Proyecto PySudoku:

Implementar una aplicación de Sudoku en Python, en la cual podamos validar si la resolución de un tablero es válida o no es válida.

Nuestra aplicación debe generar tableros de sudoku con distintas dificultades.

Nuestra aplicación debe mostrarnos ayuda durante la resolución del tablero mostrándonos, cuando un número ingresado es incorrecto si ya existe en la misma fila, columna o cuadrante

Nuestra aplicación debe ofrecernos la opción de mostrar pistas, la cual nos indicará si los numeros ingresados son correctos o incorrectos validando el tablero.

Nuestra aplicación nos permitirá llevar un registro de los mejores puntajes obtenidos al resolver el tablero en sus diferentes dificultades.

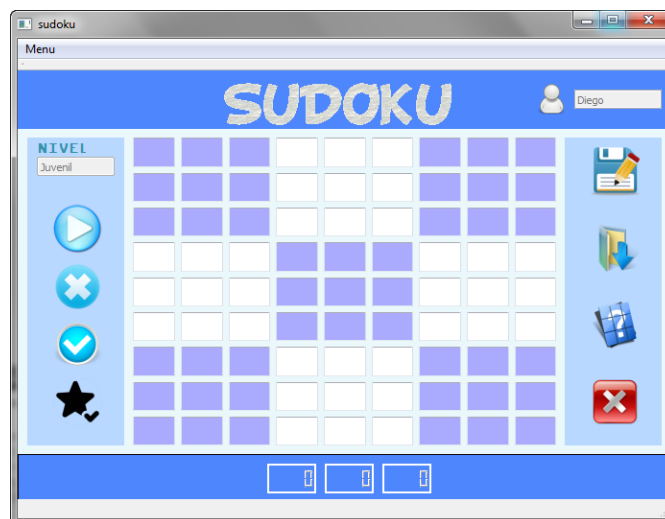


Figura 4.2: Tablero Sudoku

4.2. Herramientas Utilizadas en el Desarrollo del Proyecto:

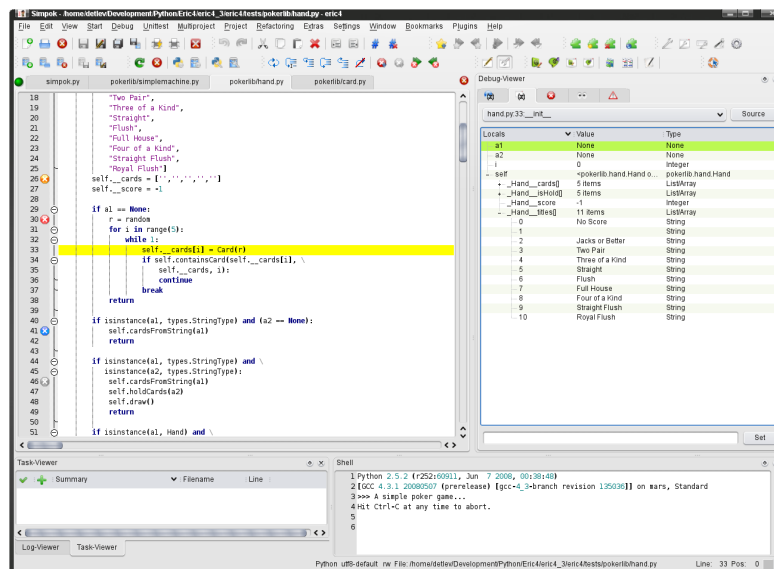


Figura 4.3: Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License, que es compatible con la Licencia pública general de GNU a partir de la versión 1.1, e incompatible en ciertas versiones anteriores.

4.3. GitHub:



Figura 4.4: git

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código se almacena de forma publica, aunque tambie puede hacer de forma privada, creando una cuenta de pago.

4.4. Doxygen:

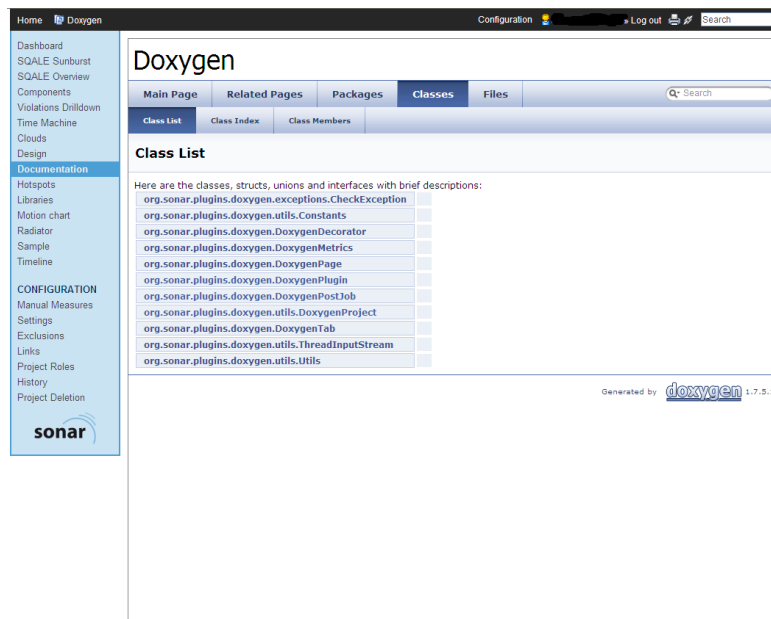


Figura 4.5: doxygen

Doxygen es un generador de documentación para C++, C, Java, Objective-C, Python, IDL (versiones Corba y Microsoft), VHDL y en cierta medida para PHP y D. Dado que es fácilmente adaptable, funciona en la mayoría de sistemas Unix asmo en Windows y Mac OS X. La mayor parte del código de Doxygen esta escrita por Dimitri van Heesch. Doxygen es un acrónimo de dox(document) gen(generator), generador de documentación para codigo fuente. Varios proyectos como KDE usan Doxygen para generar la documentacion de su API. KDevelop incluye soporte para Doxygen.

4.5. Objetivos:

Crear una aplicación de sudoku en Python.

Utilizar Python como principal herramienta de desarrollo.

Encontrar las ventajas que ofrece Python en comparación a otros lenguajes antes utilizados.

Implementar la aplicación con la metodología Orientada a objetos en Python.

Utilizar GIT como herramienta de control de versionamiento.

Utilizar Doxygen para documentacion del código.

4.6. Desarrollo:

El desarrollo de este segundo proyecto se realizó entre dos integrantes del grupo, empezando por crear la interfaz de la aplicación, que se logró hacerlo importando la interfaz utilizada en QT a Python, luego de esto se procedio a implementar la funcionalidad investigando la sintaxis del Lenguaje para el paso de la funcionalidad de QT a Python.

El trabajo fue dividido entre los dos integrantes del grupo, Carlos Ramírez desarrollo parte de la funcionalidad de las pistas, corrección ingame del sudoku de los números ingresados en el tablero por medio del evento textChanged y de la opción de dar pista que permitía mostrar al usuario que valor ingresado era correcto y cual no lo era antes de terminar de llenar el tablero, optimizar ciertas partes del código como el momento de generar tableros y la documentación de la aplicación.

José Vélez desarrollo el iniciar partida el cual generaba tableros a partir de una plantilla otra de las funcionalidades implementadas fue el cargar y guardar partida, debidamente encriptado con operaciones de desplazamiento y operaciones aritmeticas al código ASCII de los valores del tablero de sudoku. También se encargo de implementar los puntajes que se calculaba en base al tiempo tomado en resolver el juego.

4.7. Conclusiones:

La experiencia en Python, como herramienta de trabajo fue satisfactoria debido a que se orientaba a objetos tenia similitudes al lenguaje java, el inconveniente fue el poco tiempo para aprender a manejar la herramienta, una de las ventajas encontradas es que Python es un lenguaje de código mejor estructurado por su organización por indentación, con la particularidad que no se necesita declarar el tipo de variable, entre las ventajas de Python encontramos, su compatibilidad debido a ser un lenguaje interpretado, una de las desventajas podría ser un poco menos eficiente en cuestion de tiempo de ejecución ya que al ser un lenguaje interpretado consume mas recursos que un lenguaje compilado.

4.8. Referencias:

<http://mundogeek.net/tutorial-python/>

<http://docs.python.org/2/tutorial/>

<http://www.python.org/doc/>

Capítulo 5

Proyecto 3

El proyecto de Analizador Léxico desarrollado en Haskell



Figura 5.1: haskell

El objetivo del tercer Proyecto del Curso de Lenguajes de Programación fue desarrollar un Analizador Léxico de Código C, en Haskell un Lenguaje de programación funcional, el cual es completamente diferente a cualquier otra herramienta utilizada anteriormente. Uno de los Objetivos es aprovechar al máximo esta herramienta de desarrollo funcional, implementando nuestro Analizador de manera recursiva.

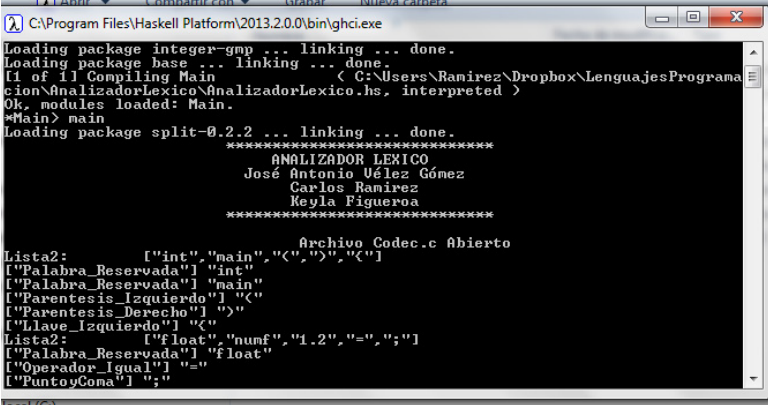
5.1. Requerimientos del Proyecto Analizador Léxico:

Implementar un Analizador Léxico, en el cual podremos analizar un código fuente escrito en c, detectar los lexemes y los tokens.

Nuestro Analizador Léxico debe generar una lista de tuplas con los lexemes encontrado en el código C.

Identificar las palabras reservadas, los separadores, los identificadores, variables, cadenas de caracteres y números en nuestro código c.

5.2. Herramientas Utilizadas en el Desarrollo del Proyecto:



```

C:\Program Files\Haskell Platform\2013.2.0.0\bin\ghci.exe
Loading package integer-gmp ... linking ... done.
Loading package base ... linking ... done.
[1 of 1] Compiling Main               < C:\Users\Ramirez\Dropbox\LenguajesProgramacion\AnalizadorLexico\AnalizadorLexico.hs, interpreted >
Ok, modules loaded: Main.
*Main> main
Loading package split-0.2.2 ... linking ... done.
*****
ANALIZADOR LEXICO
José Antonio Uélez Gómez
Carlos Ramírez
Keyla Figueroa
*****
Archivo Codec.c Abierto
Lista2:      ["int","main","<",">","{","}"]
["Palabra_Reservada"] "int"
["Palabra_Reservada"] "main"
["Parentesis_Izquierdo"] "<"
["Parentesis_Derecho"] ">"
["Llave_Izquierdo"] "{"
Lista2:      ["float","numf","1.2","=",";"]
["Palabra_Reservada"] "float"
["Operador_Igual"] "="
["PuntoyGoma"] ";"

```

Figura 5.2: haskell

Haskell es un lenguaje de programación estandarizado multi-propósito puramente funcional con semánticas no estrictas y fuerte tipificación estática. Su nombre se debe al lógico estadounidense Haskell Curry. En Haskell, una función es un ciudadano de primera clase del lenguaje de programación. Como lenguaje de programación funcional, el constructor de controles primario es la función. El lenguaje tiene sus orígenes en las observaciones de Haskell Curry y sus descendientes intelectuales. En los años 80 se constituyó un comité cuyo objetivo era crear un lenguaje funcional que reuniera las características de los múltiples lenguajes funcionales de la época, el más notable Miranda, y resolviera la confusión creada por la proliferación de los mismos. El lenguaje evoluciona rápidamente con y como los representantes actuales del estándar de facto. El último estándar semi-oficial es Haskell 98, con la intención de especificar una versión mínima y compatible del lenguaje como base para futuras extensiones y para fines educativos. Las características más interesantes de Haskell incluyen el soporte para tipos de datos y funciones recursivas, listas, tuplas, guardas y calce de patrones. La combinación de las mismas pueden resultar en algunas funciones casi triviales cuya versión en lenguajes imperativos pueden llegar a resultar extremadamente tediosas de programar.

5.3. GitHub:



Figura 5.3: git

GitHub es una forja para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el framework Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código se almacena de forma publica, aunque tambie puede hacer de forma privada, creando una cuenta de pago.

5.4. Objetivos:

Crear una Analizador Léxico en Haskell.

Entender el funcionamiento de un Analizador léxico en un lenguaje de programación.

Aprovechar Haskell como herramienta de desarrollo por su lenguaje funcional.

Implementar nuestro Analizador haciendo uso de métodos recursivos

Utilizar GIT como herramienta de control de versionamiento.

5.5. Desarrollo:

El desarrollo de este tercer proyecto se realizó entre tres integrantes del grupo, empezando por crear la función que nos permita cargar un archivo con nuestro código c, entre otros archivos como la lista de identificadores, separadores y palabras reservadas.

El trabajo fue dividido entre los tres integrantes del grupo, en este proyecto se incorporó al grupo a Keyla Figueroa.

La funcionalidad fue dividida entre Carlos Ramírez, Keyla Figueroa y Joselez.

En este proyecto como es desarrollado en un Lenguaje funcional, se colaboró mutuamente en la implementación de nuestro Analizador Léxico, investigando de diferentes fuentes el funcionamiento, formas de implementar mediante recursión el análisis del código para encontrar los lexemes y clasificarlos por tokens. El funcionamiento de nuestro analizador básicamente consiste en leer nuestro archivo de código fuente de C, y primero que todo limpiarlo de espacios y saltos de líneas, para un mejor análisis del mismo por funciones split. Luego se procedió a cargar la lista de identificadores, palabras reservadas y separadores desde archivos distintos para proceder a verificar si existen en el código C para empezar a clasificarlos por medio de tokens. Luego se procedió a identificar variables, números y cadenas de caracteres dentro de nuestro código C. Para finalmente proceder a mostrar una lista de tuplas con los lexemes y sus respectivos tokens.

5.6. Conclusiones:

La experiencia en Haskell, como herramienta de trabajo fue realmente algo novedoso por su lenguaje funcional, y como nos conlleva a pensar en soluciones de manera recursiva para un aprovechamiento de la herramienta, aparte de la no existencia de variables toda la experiencia en otros lenguajes no nos ayudó mucho para el desarrollo de este proyecto porque era una manera totalmente de programar, una perspectiva nueva. Las ventajas de Haskell es su potencial con funciones recursivas y que incluye el soporte de tipos de datos junto al manejo de listas, tuplas fue una herramienta de gran utilidad para implementar nuestro Analizador Léxico. Se cumplió con los objetivos planteados se usó Haskell como herramienta de desarrollo y entendimos el funcionamiento de un Analizador Léxico.

5.7. Referencias:

<http://aprendehaskell.es/>

<http://www.lcc.uma.es/~blas/pfHaskell/gentle/>

<http://www.haskell.org/haskellwiki/Tutorials>

<http://www.haskell.org/tutorial/>

Capítulo 6

Glosario

lexeme:

es una parte de una palabra que constituye la unidad mínima y se puede decir que es la raíz de la palabra (monema) con significado léxico.

token:

es una cadena de caracteres que tiene un significado coherente en cierto lenguaje de programación.

tupla:

es una secuencia ordenada de objetos, esto es, una lista con un número limitado de objetos (una secuencia infinita se denomina en matemática como una familia, aunque hay autores que consideran el término tupla para denominar no solo listas finitas).

semántica:

se refiere a los aspectos del significado, sentido o interpretación de signos lingüísticos como símbolos, palabras, expresiones o representaciones formales.

sintaxis:

es la parte de la gramática que estudia las reglas y principios que gobiernan la combinatoria de constituyentes sintácticos y la formación de unidades superiores a estos, como los sintagmas y oraciones gramaticales.