UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

Anexos:

BlocklyToCFacil

Presentado por Rosana Arnáiz Vicario
en febrero de 2018
Tutor Carlos Pardo Aguilar

Índice de contenido

Índice de ilustraciones

Índice de tablas

I - *PLAN DE PROYECTO*

1. Introducción

En este apartado se va a especificar el proceso de planificación temporal del proyecto y el seguimiento que se ha llevado a cabo.

A continuación se analizará la viabilidad del proyecto, desde la perspectiva económica y legal.

Si se desea, se puede consultar el proyecto en el repositorio GitHub desde el siguiente enlace *https://github.com/RosanaAV/BlocklyToChapinToCFacil*

2. Planificación temporal

Para el desarrollo del proyecto se plantea seguir una metodología ágil de tipo Scrum. El periodo de desarrollo se ha dividido en 9 sprints.

2.1. Sprint 0: 24/02/2017

Este sprint marcó el comienzo del proyecto, debido a que se elige el proyecto y se desarrolla la planificación temporal que se va a llevar a cabo hasta el día de la entrega, que es el día 15 de junio de 2017.

2.2. Sprint 1: 27/02/2017 – 05/03/2017

El objetivo de este sprint es entender el proyecto, conocer la aplicación Blockly, saber cómo funciona y decidir que herramientas se van a utilizar para el desarrollo.

También se van a fijar los objetivos del proyecto y se creará la introducción al proyecto.

2.3. Sprint 2: 06/03/2017 – 12/03/2017

El objetivo de este sprint es conocer y aprender a utilizar las herramientas que se van a usar durante el desarrollo del proyecto.

También se van a especificar los requisitos y los casos de uso.

2.4. Sprint 3: 13/03/2017 – 02/04/2017

El objetivo de este sprint es finalizar con la primera fase del proyecto, tanto con su desarrollo y programación, como con la documentación.

2.5. Sprint 4: 03/04/2017 – 16/04/2017

El objetivo de este sprint es finalizar con la segunda fase del proyecto, tanto con su desarrollo y programación, como con la documentación.

2.6. Sprint 5: 17/04/2017 – 30/04/2017

El objetivo de este sprint es finalizar con la tercera fase del proyecto, tanto con su desarrollo y programación, como con la documentación.

2.7. Sprint 6: 01/05/2017 – 14/05/2017

El objetivo de este sprint es finalizar con la cuarta fase del proyecto, tanto con su desarrollo y programación, como con la documentación.

2.8. Sprint 7: 15/05/2017 – 28/05/2017

El objetivo de este sprint es finalizar con la quinta fase del proyecto, tanto con su desarrollo y programación, como con la documentación.

2.9. Sprint 8: 29/05/2017 – 11/06/2017

El objetivo de este sprint es terminar con el proyecto, finalizar con alguna tarea que no se haya terminado, crear la documentación final, con las documentaciones creadas anteriormente y los nuevo archivos que habría que crear.

También se van a crear los anexos de especificación de diseño, documentación técnica de programación y documentación se usuario.

2.10. Sprint 9: 12/06/2017 – 15/06/2017

Los objetivos de este sprint son maquetar el proyecto, imprimirlo y entregarlo.

3. Seguimiento

Debido a motivos personales no se pudo respetar las fechas fijadas en la planificación inicial del proyecto, pero sí que se han ido siguiendo la planificación del desarrollo en 9 sprints.

3.1. Sprint 0

Como se ha dicho en la planificación del proyecto, el día 24 de febrero del 2017, es la fecha de asignación del proyecto y es cuando se desarrolla la planificación temporal que debe llevar el proyecto.

3.2. Sprint 1

En el desarrollo de este sprint se conoció la herramienta Blockly, aprendiendo como funciona y visualizando los archivos fuentes de la herramienta.

En este momento se decidió desarrollar la aplicación como una extensión de la aplicación Blockly, siguiendo el modelo de programación para la conversión de código que utiliza Blockly para los otros lenguajes de la aplicación, programando en Python y JavaScript.

Se especificaron los objetivos del proyecto y se redactó una introducción.

3.3. Sprint 2

En el desarrollo de este sprint se decidió cambiar de herramientas para la programación de la aplicación, ya no se utilizarían los lenguajes Python y JavaScript, debido a que siguiendo el modelo anterior, nuestro conversor generaría en el código variables globales, y no variables locales, así que se decidió desarrollar con el analizador léxico y sintáctico Flex/Bison.

Se especificaron los requisitos del proyecto y los casos de uso.

3.4. Sprint 3

En el desarrollo de este sprint, se empezó con la primera fase del proyecto con Flex/Bison.

En la primera fase o nivel del proyecto se quiere generar una aplicación que convierta programas que muestran un texto por pantalla, y el código de esa fase.

Se empezó a desarrollar el proyecto con Flex/Bison, pero se decidió probar con el analizador léxico y sintáctico Javacc ya que no se avanzaba, y resultó más sencillo de utilizar, por lo que se volvió a cambiar de herramienta.

Este sprint fue uno de los más largo en tiempo debido a que se tuvieron que instalar dos tipos de herramientas diferentes, aprender a utilizarlas y empezar con ambas el desarrollo del proyecto.

3.5. Sprint 4

Este sprint corresponde a el desarrollo de la segunda fase del proyecto, ampliar la aplicación para que los programas puedan incluir el uso de variables, así como su documentación.

3.6. Sprint 5

Este sprint corresponde a el desarrollo de la tercera fase del proyecto, ampliar la aplicación para que los programas puedan hacer operaciones aritméticas, así como su documentación.

3.7. Sprint 6

Este sprint corresponde a el desarrollo de la cuarta fase del proyecto, ampliar la aplicación para que los programas puedan utilizar funciones, así como su documentación.

3.8. Sprint 7

Este sprint corresponde a el desarrollo de la quinta fase del proyecto, ampliar la aplicación para que los programas puedan incluir operaciones condicionales if e if-else, así como su documentación.

Este sprint también fue largo en tiempo debido a que se localizaron fallos de las fases anteriores y se solucionaron, como por ejemplo, la declaración de las variables o el correcto posicionamiento de los paréntesis en el conversor.

3.9. Sprint 8

Durante este sprint se finalizó con el desarrollo y generación de la documentación de la memoria y los anexos del proyecto, además de la creación del archivo .jar y de la modificación del archivo make.

Este sprint también fue largo en tiempo, debido a que había que terminar de completar la documentación del proyecto.

3.10. Sprint 9

En este sprint, se maqueta el proyecto, se imprime y se entrega el día 14 de febrero de 2018.

4. Estudio de viabilidad

En el siguiente apartado se estudian los costes del desarrollo del proyecto.

4.1. Viabilidad económica

4.1.A. Costes de personal

Este proyecto se puede llevar a cabo por un desarrollador empleado a tiempo completo durante cuatro meses. Se considera el siguiente salario:

Concepto	Coste
Salario mensual neto	1.000 €
Retención IRPF (19%)	371,82 €
Seguridad social (29,9%)	585,12 €
Salario mensual bruto	1.956,95 €
Total 4 meses	**7.827,80 €**

Tabla 1: Costes de personal

4.1.B. Costes de hardware

Como único coste hardware se considera el ordenador portátil, con una amortización de 4 años.

Concepto	Coste
Ordenador portatil	600 €
Amortización mensual	12,50 €
Total 4 meses	**50 €**

Tabla 2: Costes de hardware

4.1.C. Costes varios

Resto de costes del proyecto.

Concepto	Coste
Memoria impresa y cartel	50 €
Alquiler oficina	400 €
Internet	120 €
Total 4 meses	**570 €**

Tabla 3: Costes varios

4.1.D. Costes totales

Suma de los costes del proyecto.

Concepto	Coste
Personal	7.827,80 €
Hardware	50 €
Varios	570 €
Total 4 meses	**8.447,80 €**

Tabla 4: Costes totales

4.2. Viabilidad legal

Se utiliza en el desarrollo del proyecto software externo con licencias propias, que influye sobre el apartado legal del proyecto:

- JavaCC: JavaCC está sujeto a la licencia BSD(Berkeley Softwsre Distribution)[1] de código abierto.
- Makefile: Makefile está sujeto a licencia GLP[2]

II - Especificación de requisitos

1. Introducción

En este apartado se definen los propósitos y requerimientos de la aplicación.

2. Objetivos generales

El principal objetivo del proyecto es ayudar a iniciarse en la programación a las personas que acaban de empezar el grado.

Para ello hay que crear un programa que lea un archivo xml generado desde la programación de Blockly y genere un código compilable y ejecutable en el lenguaje creado llamado C_fácil

3. Catálogo de requisitos

3.1. Requisitos funcionales

La aplicación deberá cumplir los siguientes requisitos funcionales:

- RF-1.- Se debe poder trabajar en Blockly, bien en modo local o en modo remoto.
- RF-2.- Si se va a trabajar con Blockly en modo remoto, se debe tener acceso a Internet.
- RF-3.- La aplicación debe convertir, mostrar y ejecutar un archivo xml generado desde Blockly en el lenguaje C_facil.
- RF-4.- La aplicación debe funcionar con diferentes bloques de Blockly.
  - RF-4.1.- Debe funcionar con bloques de mostrar texto por pantalla.
  - RF-4.2.- Debe funcionar con bloques de variables locales.
  - RF-4.3.- Debe funcionar con bloques de operaciones aritméticas.
  - RF-4.4.- Debe funcionar con bloques de operaciones condicionales if e if-else.
  - RF-4.5.- Debe funcionar con bloques de funciones.

3.2. Requisitos no funcionales

Además de los requisitos funcionales establecidos, la aplicación deberá cumplir los siguiente requisitos no funcionales:

- RNF-1.- Facilidad de uso: La aplicación debe de ser intuitiva y fácil de utilizar.
- RNF-2.- La aplicación debe de permitir la incorporación de nuevos módulos.

4. Especificación de requisitos

4.1. Diagrama de casos de uso

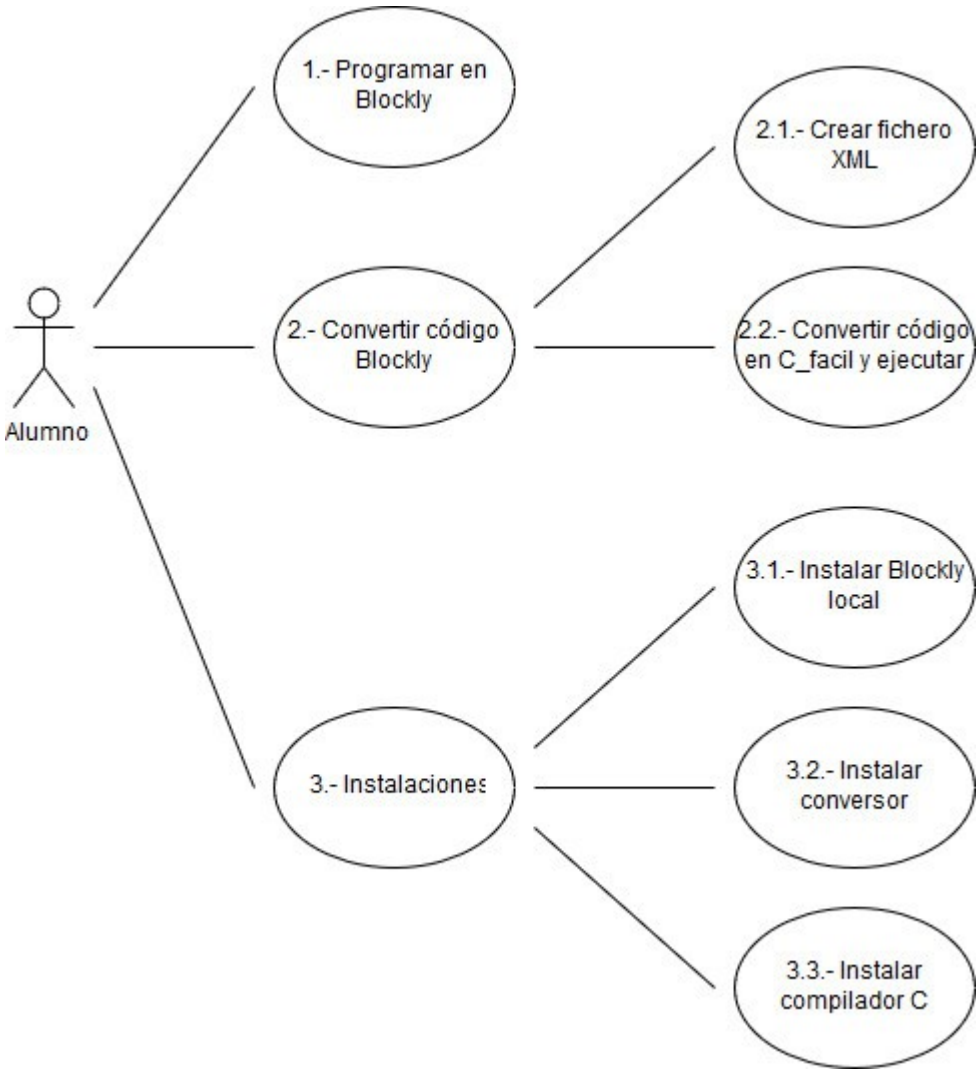

Ilustración 1: Diagrama casos de uso

4.2. Actores

Solo interatuará con el sistema un único actor, que se corresponde con el alumno.

4.3. Casos de uso

4.3.A. CU-1. – Programar en Blockly

Caso de uso	1.- Programar en Blockly
Versión	1.0.
Autor	Rosana Arnáiz Vicario
Requisitos	RF-1.- Se debe poder trabajar en Blockly, bien en modo local o en modo remoto. RF-2.- Si se va a trabajar con Blockly en modo remoto, se debe tener acceso a Internet.
Descripción	El usuario crea el programa deseado desde la aplicación Blockly uniendo diferentes bloques.
Precondiciones	Abrir la aplicación Blockly desde un navegador web.
Acciones	1.- Se unen bloques de la aplicación para generar el programa deseado. 2.- Ejecutar en Blockly para comprobar si el programa está correcto. 3.- Ver el código generado del programa en los diferentes lenguajes existentes. 4.- Guardar programa Blockly. - Generar link con el programa y guardar la URL.
Postcondiciones	Ninguna.
Excepciones	- Utilizar Blockly en modo local, véase caso de uso 3.1. - Excepción acción 1: Recuperar un programa anterior desde una URL.
Importancia	Alta

Tabla 5: Caso de uso 1: Programar en Blockly

4.3.B. CU-2.1. – Crear fichero XML

Caso de uso	2.1.- Crear fichero XML
Versión	1.0.
Autor	Rosana Arnáiz Vicario
Requisitos	RF-1.- Se debe poder trabajar en Blockly, bien en modo local o en modo remoto. RF-2.- Si se va a trabajar con Blockly en modo remoto, se debe tener acceso a Internet.
Descripción	El usuario debe poder crear un fichero .xml con el código generado del programa en Blockly.
Precondiciones	Tener abierto un programa generado en Blockly, véase caso de uso1.
Acciones	1.- Desde la pestaña XML de Blockly, seleccionar el todo el código generado y copiarlo. 2.- Pegar código en un fichero de texto y guardar con nombre deseado y extensión *.xml
Postcondiciones	Ninguna.
Excepciones	Ninguna.
Importancia	Alta

Tabla 6: Caso de uso 2.1: Crear fichero XML

4.3.C. CU-2.2. - Convertir código en C_fácil y ejecutar

Caso de uso	2.2.- Convertir código en C_facil y ejecutar
Versión	1.0.
Autor	Rosana Arnáiz Vicario
Requisitos	RF-3.- La aplicación debe convertir, mostrar y ejecutar un archivo xml generado desde Blockly en el lenguaje C_facil. RF-4.- La aplicación debe funcionar con diferentes bloques de Blockly.
Descripción	El usuario debe de poder convertir el código generado por Blockly en código C_facil y ejecutarlo directamente.
Precondiciones	Tener creado código XML, véase caso de uso 2.1.
Acciones	1.- Abrir la terminal 2.- Moverse hasta la carpeta donde se encuentran los ficheros. 3.- Se ejecuta desde la consola el make con la opción “convertiryejecutar” y el nombre del fichero *.xml. 4.- Se ejecuta el código C_facil generado.
Postcondiciones	
Excepciones	Excepción acción 3: Convertir código en C_facil 3.- Se ejecuta desde la consola el make con la opción “conversor” y el nombre del fichero *.xml. 4.- Se visualiza el código C_facil generado.
Importancia	Alta

Tabla 7: Caso de uso 2.2: Convertir código en C_facil y ejecutar

4.3.D. CU-3.1. - Instalar Blockly local

Caso de uso	3.1.- Instalar Blockly local
Versión	1.0.
Autor	Rosana Arnáiz Vicario
Requisitos	Ninguno.
Descripción	Descargar e instalar Blockly en modo local para poder utilizarlo sin conexión a la red.
Precondiciones	No haber realizado la instalación anteriormente.
Acciones	1.- Abrir en un navegador la siguiente página de Blockly https://developers.google.com/blockly/guides/get-started/web 2.- Descargar el código zip del apartado “Get the code”. 3.- Descomprimir el zip. 4.- Buscar en la carpeta la ruta demos\code el archivo index.html y abrirlo. 5.- Programar en Blockly, véase caso de uso 1.
Postcondiciones	Ninguna.
Excepciones	Ninguna.
Importancia	Media

Tabla 8: Caso de uso 3.1: Instalar Blockly local

4.3.E. CU-3.2. - Instalar conversor

Caso de uso	3.2.- Instalar conversor
Versión	1.0.
Autor	Rosana Arnáiz Vicario
Requisitos	Ninguno.
Descripción	Instalar el conversor en el equipo para poder utilizarlo.
Precondiciones	No haber realizado la instalación anteriormente.
Acciones	1.- Abrir terminal 2.- Comprobar si la versión de java es compatible con el comando “java -version” 3.- Si no existe, instalar java con el comando “sudo apt-get install default-jre” 4.- Descargar el archivo conversor.jar. 5.- Copiar el archivo en la ruta deseada.
Postcondiciones	Ninguna.
Excepciones	Ninguna.
Importancia	Alta

Tabla 9: Caso de uso 3.2: Instalar conversor

4.3.F. CU-3.3. - Instalar compilador C

Caso de uso	3.3.- Instalar compilador C
Versión	1.0.
Autor	Rosana Arnáiz Vicario
Requisitos	Ninguno.
Descripción	Descargar e instalar el compilador C para la compilación del código C_facil.
Precondiciones	No haber realizado la instalación anteriormente.
Acciones	1.- Abrir terminal 2.- Ejecutar la sentencia: “sudo apt-get install gcc”
Postcondiciones	Ninguna.
Excepciones	Ninguna.
Importancia	Alta

Tabla 10: Caso de uso 3.3: Instalar compilador C

III - Especificación de diseño

1. Introducción

En este anexos se define el diseño que se ha usado para llevar a cabo el desarrollo del proyecto.

La ejecución de la aplicación no requiere del almacenamiento de datos, sólo de la creación de ficheros.

2. Diseño procedimental

En este apartado se muestra un diagrama de flujo[3] con el funcionamiento de los procesos que se llevan a cabo en la aplicación.

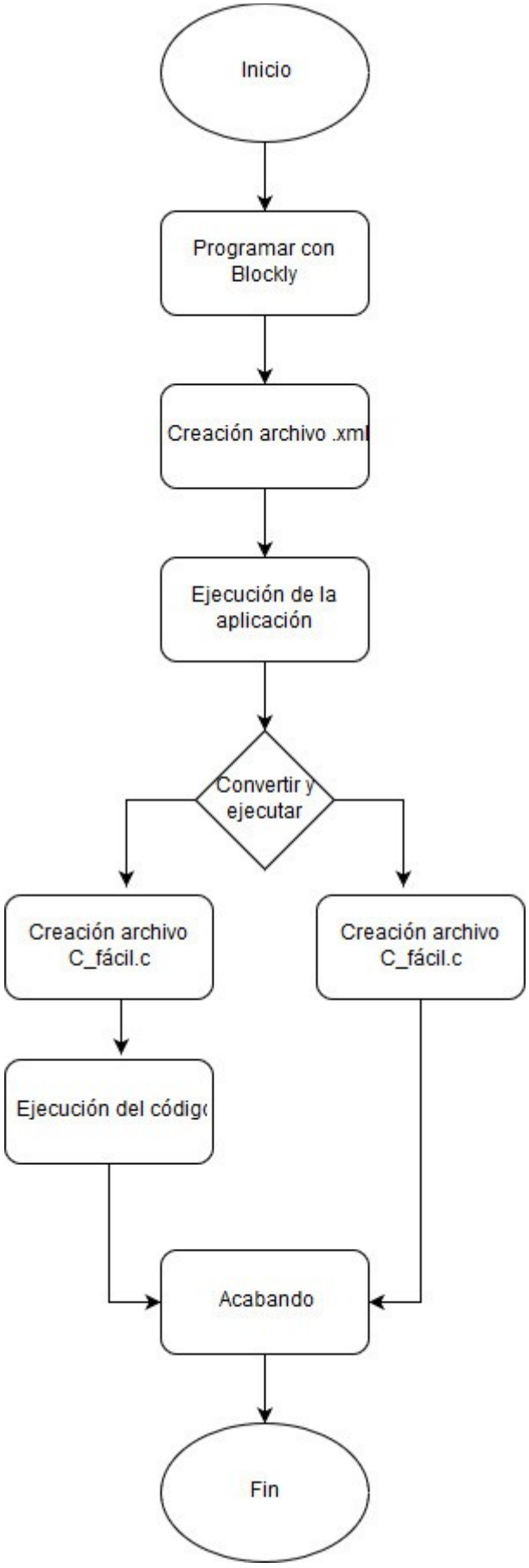

Ilustración 2: Diagrama de flujo

3. Diseño arquitectónico

La aplicación del conversor se encuentra en un fichero jar.[4]

El fichero llamado conversor.jar esta compuesto de los siguientes ficheros:

- El archivo MANIFEST.MF en la carpeta META-INF dentro del jar, que incluye el nombre de la clase principal del jar, la clase conversor que es la que incluye el main.
- Los archivos .java generados al pasar el comando “javacc conversor.jj”
  - conversor.java
  - conversorConstants.java
  - conversorTokenManager.java
  - ParseException.java
  - SimpleCharStream.java
  - Token.java
  - TokenMgrError.java
- Los archivos .class generados al pasar el comando “javac conversor.java”
  - conversor.class
  - conversorConstants.class
  - conversorTokenManager.class
  - ParseException.class
  - SimpleCharStream.class
  - Token.class
  - TokenMgrError.class

IV - Documentación técnica de programación

1. Introducción

En este anexo se describe la documentación técnica de programación.

2. Estructura de directorios

BlocklyToChapinToCFacil es el directorio general del proyecto. En el se encuentran las siguientes carpetas:

- Documentación: contiene la documentación del proyecto, tanto la memoria como los anexos.
- Fuentes: contiene la aplicación del proyecto con el nombre conversor.jar, y la configuración del c_facil
- Pruebas: contiene las pruebas que se han realizado a la aplicación.
- Seguimiento: contiene los cinco niveles en los que se ha ido programando desarrollando el proyecto de manera incremental.

3. Manual del programador

Este apartado se corresponde con una guía para desarrolladores que continuen con el proyecto.

Este proyecto está desarrollado en un sistema operativo Windows, utilizando el analizador JavaCC.

3.1. JavaCC

Para poder utilizar JavaCC en el equipo, lo primero es tener descargado el jdk de Java, que se puede descargar desde el siguiente enlace http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Después hay que introducir el jdk al path de las variables de entorno.

Después hay que descargar el JavaCC, que se puede descargar desde el siguiente enlace https://javacc.org/download, y después meterlo en el path de las variables de entorno.

Para poder programar con ello, hay que abrir un fichero de texto, programar con la sinaxis de JavaCC, y guardar el fichero con extensión .jj.

4. Compilación y ejecución del proyecto

La compilación de la aplicación se hace desde la consola de comandos, situándonos en la misma carpeta donde se encuentra el fichero .jj, con los siguientes comandos:

javacc conversor.jj

javac conversor.java

Con el primer comando se generan los archivos .java y con el segundo los archivos .class.

Para la ejecución del proyecto, lo primero que hay que hacer es programar en Blockly el programa deseado. Después, con el archivo .xml en la misma ruta que la aplicación conversor, se ejecuta el siguiente comando:

java conversor < fichero.xml > ficheroCF.c

Se genera un fichero .c en la misma ruta con el código en C_facil.

Una vez compilado el código de la aplicación y generado todos los archivos correspondientes, se crea un archivo llamado conversor.jar donde se almacenan todos los archivos.

Para crear el archivo .jar, nos situamos en la carpeta donde se han generado los archivos y se crea un nuevo archivo de texto llamado manifiesto.txt donde se almacena la clase principal de la aplicación, que es el conversor. En esta clase se escribe lo siguiente:

"Main – class: conversor"

Una vez que esta creado el manifiesto, desde la consola de comandos se crea el .jar con el siguiente comando:

jar cvfm conversor.jar manifiesto.txt *.java *.class"

Para ejecutar la aplicación, desde la consola de comandos se escribe el siguiente comando:

java -jar conversor.jar < fichero.xml > ficheroCF.c

Para que su utilización sea más sencilla, se ha incluido la ejecución del conversor en el archivo makefile utilizado en la asignatura.

5. Pruebas del sistema

En el apartado Pruebas se encuentran 7 pruebas diferentes:

5.1. Prueba1

Esta prueba corresponde con el uso de la aplicación para programas que muestran texto por pantalla.

Esta prueba da el resultado esperado.

5.2. Prueba 2

Esta prueba corresponde con el uso de la aplicación para programas que usan variables.

Esta prueba da el resultado esperado.

5.3. Prueba 3

Esta prueba corresponde con el uso de la aplicación para programas que usan operaciones aritméticas.

Esta prueba da el resultado esperado.

5.4. Prueba 4

Esta prueba corresponde con el uso de la aplicación para programas que utilizan funciones.

Esta prueba da el resultado esperado.

5.5. Prueba 5

Esta prueba corresponde con el uso de la aplicación para programas que utilizan operaciones condicionales if-else.

Esta prueba da el resultado esperado.

5.6. Prueba 6

Esta prueba corresponde con el uso de la aplicación para programas con recursividad.

Esta prueba da el resultado esperado.

5.7. Prueba 7

Esta prueba corresponde con el uso de la aplicación para programas con bloques que no se han contemplado en el conversor.

Esta prueba no muestra ningún mensaje de error, el código se crea erróneamente.

V - Documentación de usuario

La documentación de usuario se encuentra en el siguiente enlace https://blocklytocfacil.wordpress.com/ .

Se ha decidido utilizar un recurso externo para almacenar la documentación de usuario debido a que esta aplicación esta especialmente dirigida a alumnos del grado, y tener el manual de uso en una página web facilitaría su visualización.

VI - Referencias

[1] Licencia BSD, online: https://es.wikipedia.org/wiki/Licencia_BSD

[2] Licencia GLP, online: http://www.gnu.org/

[3] Diagrama de flujo, online: https://es.wikipedia.org/wiki/Diagrama_de_flujo

[4] Archivo JAR, online: https://es.wikipedia.org/wiki/Java_Archive

Impreso en Burgos el miércoles, 14 de febrero de 2018