Proyecto Integrador

Unidades de Aprendizajes

Unidad(es) programática(s) enmarcada(s) en esta actividad:
Unidad 2. Análisis léxico
Unidad 3. Análisis sintáctico

Resultado de Aprendizaje

El resultado de aprendizaje que se pretende lograr con este práctico es:
Desarrolla un programa que realice las primeras fases de un compilador utilizando herramientas típicas

Presentación

Un compilador es un programa que traduce los programas escritos en un lenguaje de programación (lenguaje fuente) a otro lenguaje (lenguaje objeto). El lenguaje fuente, generalmente es un lenguaje de alto nivel y el objeto, un lenguaje de bajo nivel.
El desafío de este proyecto integrador consiste en desarrollar un programa traductor que a partir de un programa escrito en el lenguaje *ArduCompi* (lenguaje ficticio) obtenga un programa equivalente en lenguaje para la plataforma Arduino[16].
Para ello, deberá desarrollar las dos primeras fases de un compilador y generar un programa equivalente en el lenguaje objeto (LO). En este proyecto se aplicarán y desarrollarán los conocimientos adquiridos durante el cursado de la asignatura.
Básicamente deberá construir una aplicación que:

a) Analice instrucciones de un programa escrito en el lenguaje fuente asignado.
b) Determine e informe si el programa es sintácticamente correcto.
c) En caso positivo, deberá generar un archivo con el programa traducido y válido para el lenguaje objeto. De lo contrario, informará los errores encontrados.

Como parte de las condiciones de regularización (ver propuesta de cátedra), deberá presentar dos avances del trabajo, antes del plazo estipulado.

Descripción de las actividades o consignas de trabajo

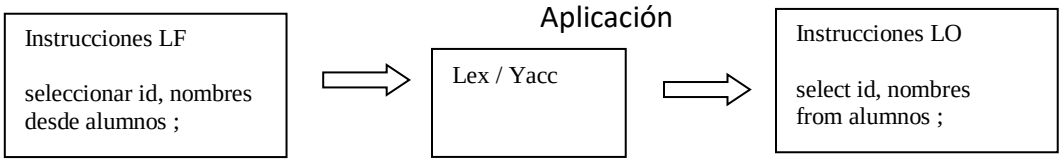
Este proyecto consiste en implementar las primeras dos fases de un compilador y generar un programa equivalente en otro lenguaje.
Puede desarrollarlo en grupo de hasta dos personas, como máximo.
De manera sintética, los pasos que deberán seguir son:

1) Se construirá el analizador léxico usando **Lex** y el analizador sintáctico usando ***Yacc***.

[16] El lenguaje de programación para Arduino es una adaptación de C++.

2) La aplicación deberá leer, desde un archivo, instrucciones del lenguaje asignado y devolver otro archivo con un programa válido para Arduino.

Ejemplo:



3) Si en el proceso de análisis se encuentran errores (léxicos y/o sintácticos), deberá informar el número de línea donde se produjeron y una descripción del error encontrado.
4) Si en el proceso de análisis no se encuentran errores, deberá crear un archivo que contenga el programa equivalente en el lenguaje objeto.

Observaciones:

- Cada grupo tiene que implementar los analizadores sólo para el conjunto de instrucciones asignado.
- Antes de las entregas se recomienda consultar y presentar tokens y la gramática para correcciones.

Avances

Los dos avances que deberán presentar son de carácter obligatorio y condicionantes para obtener la regularidad de la materia (ver propuesta de cátedra). Los alumnos que no presenten alguno quedarán en condición de libres. Estas entregas se deberán realizar a través de la plataforma en el espacio destinado a tal fin dentro del plazo establecido. **No se recibirán avances fuera de ese plazo.**

- ***26 de Septiembre:*** Entrega de ***lista de tokens*** correspondientes al lenguaje fuente asignado.

  Esta entrega consiste en un pequeño informe donde se muestre la definición de los tokens como expresiones regulares y, en caso que considere necesario, una breve explicación.

- ***31 de Octubre:*** Entrega de ***gramática del lenguaje*** fuente asignado.

  Esta entrega debe contener la gramática que permita generar programas en el lenguaje asignado (notación BNF o similar) junto con los tokens corregidos en base a las observaciones recibidas (si las hubiere) y fueron definidos en la entrega anterior.

Como actividad **optativa** se propone realizar pruebas con un robot Arduino.

- ***7 de Noviembre***: Quienes hayan avanzado con el programa traductor podrán generar archivos con código del lenguaje objeto (Arduino) para cada programa (LF) entregado por la cátedra. Esta traducción se copiará al robot y podrán probarlo en un robot real.

Informe final

La presentación final del proyecto **mínimamente** deberá incluir:

- Introducción
  - Se sugiere una breve introducción donde quedará reflejado el trabajo a realizar.
- Tokens y gramática del lenguaje
  - Se deberán incluir el listado de tokens y gramática corregidos según las indicaciones de las docentes. Además, puede enriquecer este apartado con lo que considere importante reflejar.
- Implementación
  - Se deberá indicar cómo implementó la gramática y los tokens haciendo uso de **Lex** y **Yacc**. Por ejemplo, puede incluir una breve descripción de las funciones utilizadas y cualquier otra información que considere relevante.
  - Detallar los problemas surgidos (si los hubo) durante el desarrollo del proyecto y la manera en que fueron subsanados.
- Conclusión
  - Realizar una conclusión del proyecto indicando puntos fuertes y débiles.
- Bibliografía
  - Se deberán indicar todas las fuentes bibliográficas consultadas siguiendo las normas APA.
- Códigos fuentes
  - Se incluirán los códigos fuentes de la aplicación.
- Ejecución de la aplicación
  - Este apartado será de suma importancia para evaluar el comportamiento de su aplicación. Por esto, deberá incluir las capturas de pantalla que ilustren el funcionamiento de la aplicación y cómo se informan los errores léxicos y sintácticos.

*Recuerde que esta estructura es la **mínima** a presentar pudiendo ser **mejorada y/o modificada** en caso que lo considere necesario. En las entregas no olvidar incluir el número de grupo asignado.*

Este informe final junto con los archivos fuentes y al menos tres archivos de ejemplos deberá ser enviado por mail a las docentes hasta **48hs** antes del turno de examen.
Entre los archivos de ejemplos deberá presentar:

- un programa con errores léxicos
- un programa con errores sintácticos
- un programa válido y su correspondiente traducción al lenguaje objeto

En cada uno de los programas indicar dónde se encuentran los errores y de qué tipo son, comentando el código.

El día del examen final, cada alumno responderá las preguntas de las docentes sobre el desarrollo del trabajo realizado. Si bien el trabajo es grupal, la nota individual dependerá del desarrollo del proyecto y la defensa oral del mismo.

La nota de este trabajo integrador realiza su aporte a la nota final de la materia, según lo indicado en la propuesta de cátedra.

Lenguaje Fuente Grupo 16:

La estructura del lenguaje fuente asignado es:

Principio y fin

El programa comienza con la palabra reservada ***PEMP*** y termina con ***PTER.***

Carácter de fin de línea

Cada línea de este lenguaje finaliza con el carácter **.**

Librerías externas

Este lenguaje permite incluir librerías externas. Se deben definir al inicio del archivo siguiendo la estructura: ***INCLUIR(nombreDeLibreria.Extension)***. Se pueden incluir varias una en cada línea.

Tipos de datos admitidos

Sólo se admiten los siguientes tipos de datos: ***entero (int), texto (String), decimal (float) y lógico (bool)***.

Definir variables

Para definir variables se utiliza ***VBLE(nombre1 tipo1)*** donde tipo puede ser alguno de los tipos admitidos.

Asignación de valores

Para asignar valores a una variable se utiliza el símbolo **=** seguido por una ***variable*** o un ***valor*** de algunos de los tipos admitidos.

Funciones y procedimientos

Las funciones tienen la forma de ***DEFF nombre(arg1:tipo1,arg2:tipo2,…,argn:tipon): tipo_retorno.*** En el caso que sea un procedimiento, su definición sería: ***DEFP nombre(arg1:tipo1,arg2:tipo2,…,argn:tipon).***
El procedimiento que se utiliza para definir los pines será ***DEFPI(tipo:numero)*** donde ***número*** indica el PIN que se va a utilizar cuyo ***tipo*** puede ser de salida (***PINOU***) o de entrada (***PININ***).

Comentarios

Los comentarios de línea se empiezan con ***//**** *y los de bloque* se encierran entre ***//** **//***.

Estructuras de control

En este lenguaje las únicas estructuras son:

- ***SI () () SINO ()***
- ***SI ()***
- ***DURANTE () []***

Palabras reservadas y funciones establecidas

Las palabras reservadas se escriben en MAYÚSCULAS. Además de las ya mencionadas en los apartados anteriores, otras palabras reservadas son:

- ***AVAN():*** indica que el robot puede avanzar
- ***RETRO():*** indica que el robot retrocede
- ***GIRAI():*** indica que el robot dobla en sentido anti horario
- ***GIRAD():*** indica que el robot doble en sentido horario
- ***WAIT(tiempo):*** indica que el robot espera un ***tiempo*** determinado
- ***STOP():*** indica que el robot se detiene

Lenguaje Objeto

El lenguaje objeto seleccionado será Arduino. El lenguaje de programación de Arduino está basado en C++ aunque se pueden utilizar otros lenguajes. Su referencia se encuentra en http://arduino.cc/en/Reference/HomePage.
Como se puede apreciar en la Figura 1, la estructura básica de programación de Arduino es bastante simple y divide la ejecución, básicamente, en dos partes: setup y loop. **Setup()** constituye la preparación del programa y **loop()** es la ejecución.

FIGURA 1: EJEMPLO DE PROGRAMA ARDUINO

El programa traducido deberá ser generado en un archivo de texto con la estructura básica de Arduino. Cada programa se denomina sketch y su extensión es **.ino**. Para esta estructura básica se debe tener en cuenta:

- Al inicio del sketch se pueden definir variables globales e incluir librerías

  ```
  #include <nombre_librería>
  ```

- La sección de inicialización de pines se corresponde con el método setup que tiene la forma:

  ```
  void setup(){
        <inicialización_de_pines>
  }
  ```

  Esta ***función setup()*** se trata de la primera que se ejecuta en el programa. Esta función se ejecuta una única vez y es empleada para configurar los modos de los pines, es decir si un determinado pin es de entrada o salida (pinMode) e inicializar la comunicación serie.

- El cuerpo del programa se corresponde con el método loop que tiene la forma:

```
void loop(){
      <lista_de_instrucciones>
}
```

Esta ***función loop()*** incluye el código a ser ejecutado continuamente.

En el caso que no se haya especificado algunos operadores u otra estructura que considere necesaria para la correcta traducción, se podrán utilizar las definiciones del propio lenguaje Arduino indicando claramente en el informe la justificación de su uso.

Ejemplo de lo esperado:

Programa Fuente

```
PEMP
INCLUIR(nombreDeLibreria.Extension).
VBLE(MD1 entero).
VBLE(MD2 entero).
MD2=3.//*INICIO DE LA SECCION SETUP
DEFPI(PINOU:MD1) .
DEFPI(PINOU:MD2).
//*FIN DE LA SECCION LOOP
AVAN().
GIRAI().
STOP().
PTER
```

Programa Objeto:

```
#include < nombreDeLibreria.extension >
int MD1;
int MD2;
MD2:=3;
void setup(){
       pinMode(MD1, OUTPUT);
       pinMode(MD2, OUTPUT);
}
void loop(){
       avanzar();
       giro_izquierda();
       parar();
}
```

Archivo funciones.h

En caso que pruebe la traducción en el robot real, se incluirá una librería llamada funciones. En el archivo ***funciones.h*** se encuentran definidas aquellas funciones que permiten realizar el movimiento deseado del robot. A continuación, se detalla cada una y se muestra su implementación **solo para su conocimiento**. No deben redefinirlas sino simplemente utilizarlas.

Función **avanzar()**: Esta función permite que el robot avance. Para que esto ocurra, se debe indicar a Arduino que la rueda derecha gire en sentido horario y la izquierda en sentido anti horario. Para su conocimiento, su implementación es la siguiente:

```
void avanzar(){
    digitalWrite(MD1,LOW);
    digitalWrite(MotorDer2,HIGH);
    digitalWrite(MotorIzq1,LOW);
    digitalWrite(MotorIzq2,HIGH);
    analogWrite(PWM_Derecho,200);//Velocidad motor derecho 200
    analogWrite(PWM_Izquierdo,200);//Velocidad motor izquierdo 200
  }
```

Función **retroceder()**: Esta función permite que el robot retroceda. Para que esto ocurra, se debe indicar a Arduino que la rueda derecha gire en sentido anti horario y la izquierda en sentido horario.

```
void retroceder(){
    digitalWrite(MotorDer1,HIGH);
    digitalWrite(MotorDer2,LOW);
    digitalWrite(MotorIzq1,HIGH);
    digitalWrite(MotorIzq2,LOW);
    analogWrite(PWM_Derecho,200);//Velocidad motor derecho 200
    analogWrite(PWM_Izquierdo,200);//Velocidad motor izquierdo 200
  }
```

Función **giro_derecha()**: Esta función permite que el robot gire a la derecha. Para que esto ocurra, se debe indicar a Arduino que ambas ruedas giren en sentido horario.

```
void giro_derecha(){
    digitalWrite(MotorDer1,HIGH);
    digitalWrite(MotorDer2,LOW);
    digitalWrite(MotorIzq1,LOW);
    digitalWrite(MotorIzq2,HIGH);
    analogWrite(PWM_Derecho,200);//Velocidad motor derecho 200
    analogWrite(PWM_Izquierdo,200);//Velocidad motor izquierdo 200
  }
```

Función **giro_izquierda()**: Esta función permite que el robot gire a la izquierda. Para que esto ocurra, se debe indicar a Arduino que ambas ruedas giren en sentido anti horario.

```
void giro_izquierda(){
    digitalWrite(MotorDer1,LOW);
    digitalWrite(MotorDer2,HIGH);
    digitalWrite(MotorIzq1,HIGH);
    digitalWrite(MotorIzq2,LOW);
    analogWrite(PWM_Derecho,200);//Velocidad motor derecho 200
    analogWrite(PWM_Izquierdo,200);//Velocidad motor izquierdo 200
  }
```

Función **parar()**: Esta función detiene los motores del robot. Para que esto ocurra, se debe indicar a Arduino que ambas ruedas estén en estado LOW.

```
void parar(){
    digitalWrite(MotorDer1,LOW);
    digitalWrite(MotorDer2,LOW);
    digitalWrite(MotorIzq1,LOW);
    digitalWrite(MotorIzq2,LOW);
    analogWrite(PWM_Derecho,0);
    analogWrite(PWM_Izquierdo,0);
  }
```

Recursos y lugar de aprendizaje

Los recursos que debe utilizar para realizar este práctico son:

- Lección: Lenguajes regulares (aula virtual)
- Lección: Gramáticas regulares (aula virtual)
- Lección: Análisis léxico (aula virtual)
- Lección: Lenguajes independientes del contexto (aula virtual)
- Lección: Análisis sintáctico, Análisis sintáctico no recursivo y Análisis sintáctico ascendente (aula virtual)
- Lecciones: Lex y Yacc. Ejemplos (aula virtual)
- Material provisto por la cátedra (aula virtual)
- Bibliografía básica referida a la Unidad 2 y a la Unidad 3 (ver propuesta de cátedra)
- Documentación oficial de PLY disponible en http://www.dabeaz.com/ply/index.html

Evaluación

En el informe final se evaluarán los siguientes criterios:

		No presentado / contestado	No logrado	En proceso	Logrado	Excelente
	Porcentaje	**0**	**4**	**6**	**8**	**10**
Aplica las herramientas LEX y YACC adecuadamente para desarrollar las primeras fases de un compilador	**60%**	No entregado.	El alumno no aplica correctamente ninguna de las herramientas.	El alumno aplica las herramientas pero se evidencian errores conceptuales en su uso.	El alumno aplica las herramientas, pero sus ejecuciones contienen errores.	El alumno aplica correctamente las herramientas sin presencia de errores en sus ejecuciones.
		0	2.4	3.6	4.8	6
Elabora adecuadamente los informes de avance del proyecto	**10%**	No entregado.	El alumno no respeta las pautas para la elaboración de los informes de avance y el informe contiene muchos errores.	El alumno respeta algunas pautas para la elaboración de los informes y el informe contiene algunos errores.	El alumno respeta la mayoría de las pautas para la elaboración de los informes y/o el informe contiene pocos errores.	El alumno respeta las pautas para la elaboración de los informes y el informe no contiene errores.
		0	0.4	0.6	0.8	1

Precisión en el uso de los conceptos	**20%**	No entregado.	El alumno no utiliza de manera precisa los conceptos involucrados en el trabajo integrador.	El alumno utiliza de manera precisa algunos de los conceptos involucrados en el trabajo integrador.	El alumno utiliza de manera precisa la mayoría de los conceptos involucrados en el trabajo integrador.	El alumno utiliza de manera precisa todos los conceptos involucrados en el trabajo integrador.
		0	0.8	1.2	1.6	2
Claridad en la redacción, estilo académico, vocabulario técnico, correcta construcción gramatical, puntuación y acentuación	**10%**	No entregado.	El informe presentado contiene errores en todas las condiciones enunciadas en el criterio de evaluación.	El informe presentado contiene errores en la mayoría de las condiciones enunciadas en el criterio de evaluación.	El informe presentado contiene errores en algunas de las condiciones enunciadas en el criterio de evaluación.	El informe presentado posee claridad en la redacción, estilo académico, vocabulario técnico, correcta construcción gramatical, puntuación y acentuación.
		0	0.4	0.6	0.8	1

Tiempo estimado de realización

El tiempo estimado para completar este trabajo es:

Horas presenciales	7 hs.
Horas de trabajo autónomo	15 hs.
Tiempo total	22 hs.

Formato y fecha de entrega

El primer avance debe entregarse antes del **26 de septiembre** (hasta las 23.59hs) y el segundo antes del **31 de octubre** (hasta las 23.59hs); ambos en formato PDF a través de la plataforma. NO se recibirán entregas por e-mail. Las dudas y aclaraciones sobre el enunciado se resolverán a través del foro.

Nota: **Propiedad intelectual**

Al presentar un práctico que haga uso de recursos ajenos, se detallarán todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y su estatus legal: si la obra está protegida por copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia no impide específicamente su uso. En caso de no encontrar la información correspondiente deberá asumir que la obra está protegida por copyright.