**UNIVERSIDAD TECNOLÓGICA CENTROAMERICANA**

**FACULTAD DE INGENIERÍA**

****

**PROYECTO FINAL**

**CLASE: COMPILADORES II**

**TEMA: Manual de SmileBasic**

**INTEGRANTES:**

**EDWIN JAVIER NÚÑEZ - 10911188**

**OSCAR MEJIA - 11441233**

**SWAMY ANTUNEZ – 11141320**

**CARRERA DE ESTUDIO: INGENIERIO EN SISTEMAS**

**PROFESOR: CARLOS VALLEJO**

**FECHA DE ENTREGA: 29 de Junio**

**Tegucigalpa MDC, Honduras, Centroamérica 2018**

# Contenido

[ESPECIFICACION DEL LENGUAJE 1](#_Toc518055893)

[MANUAL TÉCNICO DE SMILEBASIC 1](#_Toc518055894)

[MANEJO DEL ARBOL (AST) 2](#_Toc518055895)

[MANUAL DE USUARIO DE SMILEBASIC 2](#_Toc518055896)

[INSTRUCCIONES 2](#_Toc518055897)

[INPUT 3](#_Toc518055898)

[LINPUT 3](#_Toc518055899)

[GOTO 3](#_Toc518055900)

[GOSUB 4](#_Toc518055901)

[Operaciones de Control 4](#_Toc518055902)

[IF 4](#_Toc518055903)

[ON 5](#_Toc518055904)

[Operadores Lógicos 5](#_Toc518055905)

[&& (AND) 5](#_Toc518055906)

[|| (OR) 6](#_Toc518055907)

[! (NOT) 6](#_Toc518055908)

[Declaración de Variables 6](#_Toc518055909)

[VAR 6](#_Toc518055910)

[INT 6](#_Toc518055911)

[DOUBLE 6](#_Toc518055912)

[STRING 6](#_Toc518055913)

[Iteraciones 7](#_Toc518055914)

[FOR 7](#_Toc518055915)

[WHILE 7](#_Toc518055916)

[REPEAT 7](#_Toc518055917)

[Funciones 7](#_Toc518055918)

[DEF 7](#_Toc518055919)

[Metodos 8](#_Toc518055920)

[SWAP 8](#_Toc518055921)

[INC 8](#_Toc518055922)

[DEC 9](#_Toc518055923)

[CONTINUE 9](#_Toc518055924)

[CONCLUSION 9](#_Toc518055925)

# ESPECIFICACION DEL LENGUAJE

El lenguaje SmileBasic se usa de las siguientes maneras:

* Usamos algunos archivos de java con clases y un main
* Con analizador léxico
* Con analizador sintáctico
* Con analizador semántico
* Pseudocódigo en archivos de prueba

# MANUAL TÉCNICO DE SMILEBASIC

El SmileBasic funciona de las siguientes maneras:

* Creamos un archivo .jflex para declarar las instrucciones que vamos a utilizar y que retornen sus símbolos.
* Creamos un archivo .cup para declarar las terminales y no terminales y así creamos el **árbol sintáctico abstracto**.
* Creamos un archivo Main.java para que llame al parser.cup pasándole el lexer.jflex y un FileReader.
* En los archivos de jflex y cup usamos las librerías de “java-cup-11b.jar”, “jflex-1.6.1.jar” y “java-cup-11b-runtime.jar” para compilar que salgan bien.
* En la consola se hace lo siguiente:

1. Usar jflex para compilar el .flex a un .java.

>java -jar [jflex.jar] [flex file]

2. Usar java y cup para compilar el cup a un .java y crear el archivo sym.

>java -jar [cup.jar] [cup file]

3. Compilar todo con javac. Mandar el classpath de cup runtime y el directorio con todos los class si es necesario. Primero el sym , después clases externas, luego el lexer, el parser y finalmente el Main.

>javac -cp '[cup-runtime.jar];.' [java file]

4. Correr el Main.class y mandar el archivo como argumento.

>java -cp '[cup-runtime.jar];.' Main [test file]

## MANEJO DEL ARBOL (AST)

La estructura del árbol se hace en el archivo parser.cup en donde se desarrollan las gramáticas para los terminales, no terminales y precedencias. Cada nodo del árbol denota una construcción que ocurre en el código fuente. Una vez construido, información adicional es agregada al AST por procesamiento subsecuente, ej., análisis [semántico](https://es.wikipedia.org/wiki/Sem%C3%A1ntico).

La construcción del árbol se hace en el archivo parser.cup con el método BuildAstAsDot(Node n) el cual concatena la representación de cadena de nodos del método NodeToDot(…) de cualquier otro tipo de datos al final del objeto y valida en caso de algún error.

# MANUAL DE USUARIO DE SMILEBASIC

Esta basado en un lenguaje BASIC. Como se sabe BASIC es de los mejores lenguajes de programación para principiantes y así aprendemos a programar desde cero. BASIC originalmente fue desarrollado como una herramienta de enseñanza. El lenguaje y sus variantes llegaron a estar ampliamente disponibles en los [microcomputadores](https://es.wikipedia.org/wiki/Microcomputador) a finales de los años 1970 y en los años 1980. ¡Tan pronto como inicie SmileBASIC, puede comenzar a programar!

## INSTRUCCIONES

**PRINT:** Instrucción en la cual se muestra impreso cualquier cadena de carácter, números, símbolos, etc. entre comillas. Si se escribe 2 PRINT y el primero se le escribe al final ‘;’ salen impreso los 2 como uno solo, pero si se le escribe ‘,’ al primero salen los dos PRINT separados, siempre en la misma línea, pero con un espacio.

EJEMPLO:

PRINT “hello”;

PRINT “world”

SALIDA:

helloworld

1. PRINT “hello”,
2. PRINT “world”

SALIDA:

hello world

## INPUT

Instrucción en la cual se le ingresa valor a la variable durante la salida (con el teclado) en espera de un valor especifico a la variable ya sea número o carácter y luego presiona ENTER. Si el número de elementos de entrada es insuficiente, se mostrará "Rehacer desde el inicio" para volver a ingresar

EJEMPLO:

1. INPUT “Valor de la base: ”;B

2 A = 4

1. C = B \* A
2. PRINT “El área de un triángulo es”; C

SALIDA:

Valor de la base: ? 6

El área de un triángulo es 24

## LINPUT

Instrucción igual a INPUT con la excepción de que sólo se le ingrese una cadena de caracteres y siempre en espera hasta que se presiona ENTER

EJEMPLO:

1. LINPUT “Nombre Completo: ”; Name$
2. PRINT “Su nombre es”; Name$

SALIDA:

Nombre Completo: ? Edwin Nuñez

Su nombre es Edwin Nuñez

## GOTO

Instrucción que hace un salto de línea de la primera hasta la última línea y siempre debe empezar con @ antes de los caracteres que se le escriba a lado. hará que este programa siempre regrese a la primera línea por lo que, para detenerlo presione ENTER.

EJEMPLO:

1. @PRIMERO
2. INPUT “Valor de la base: ”; B
3. A = 4
4. C = B \* A
5. PRINT “El área de un triángulo es”; C

6 GOTO @PRIMERO

SALIDA:

Valor de la base: ? 6

El área de un triángulo es 24

Valor de la base: ? 2

El área de un triángulo es 8

Valor de la base: ?

## GOSUB

Instrucción que llama una subrutina de acuerdo con el valor de la variable principal. En el GOSUB se le da labels hará que los comandos se ejecutarán hasta que se encuentre una instrucción **RETURN**, en cuyo punto la ejecución se transferirá a la instrucción inmediatamente posterior a la instrucción GOSUB.

EJEMPLO:

1. VAR X = 0
2. WHILE X < 10
3. GOSUB @AGREGAUNO
4. PRINT X
5. WEND
6. @AGREGAUNO
7. INC X
8. RETURN

SALIDA:

0 1 2 3 4 5 6 7 8 9

# Operaciones de Control

## IF

Instrucción para verificar el valor de una variable si cumple o no con la condición y ejecutar las instrucciones dentro del IF al final de la condición se coloca **THEN** para luego empezar a ejecutar la condición del IF si se cumple en caso contrario se va a la instrucción **ELSE** que se usa en caso que no se cumpla la condición IF para que cumpla entonces la alternativa. Si en cualquier otro caso no se cumple el IF y si necesitamos al mismo tiempo otra condición para verificar que se cumpla o no se usa **ELSEIF** al final de esto se usa **ENDIF** para marcar por terminado toda la condición IF.

EJEMPLO:

1. @PRIMERO
2. INPUT “Valor de la base: ”; B
3. IF B == 0 THEN
4. PRINT “continuar”
5. ELSEIF B <= 2 THEN
6. B = B + 1
7. ELSE
8. A = 4
9. C = B \* A
10. PRINT “El área de un triángulo es”; C
11. GOTO @PRIMERO
12. ENDIF

## ON

Instrucción que se extiende de acuerdo con el valor de la variable de control. El numero inicial seria 0. Dentro de ON llamamos a la instrucción **CASE** que se usa como punto de control para dar múltiples opciones y escoger. Al final de cada CASE se termina con un **BREAK** para terminar el ciclo de CASE. BREAK se usa también en IF FOR ... NEXT, WHILE ... WEND, REPEAT ... UNTIL. Y si ninguno de los CASES es válido se **DEFAULT** para mostrar el error**.** Al final se termina todo con la instrucción **WON**

EJEMPLO:

1 ON crud

2 CASE 1: PRINT "Crear"

3 BREAK

4 CASE 2: PRINT "Leer"

5 BREAK

6 CASE 3: PRINT "Actualizar"

7 BREAK

8 CASE 4: PRINT "Eliminar"

9 BREAK

10 DEFAULT: PRINT "Opción Invalido"

11 BREAK

12 WON

# Operadores Lógicos

## && (AND)

Instrucción que se usa en el IF para verificar mas de una condición en el IF y sólo es verdadero el resultado si se cumple ambas condiciones

EJEMPLO:

1 IF A < B && B < C THEN

2 PRINT “TRUE”

1. ELSE
2. PRINT “FALSE”

## || (OR)

Instrucción que se usa en el IF para verificar que una u otra condición en el IF sea verdadera pero si ambas condiciones no se cumplen entonces es falsa

EJEMPLO:

1 IF X == 5 || Y == 5 THEN

1. PRINT “FALSE”
2. ELSE

4 PRINT “TRUE”

## ! (NOT)

Instrucción que se usa en el IF para verificar que la condición sea NO a lo que se esta validando, o sea que si es NO la condición es verdadera y es falsa si se cumple lo contrario al NO.

EJEMPLO:

1 IF !(A == B) THEN

1. PRINT “TRUE”
2. ELSE

4 PRINT “FALSE”

# Declaración de Variables

## VAR

Instrucción que declara variables para usar. Cada variable que se utiliza debe declararse.

EJEMPLO:

1 VAR A, KG, T$

## INT

Instrucción que declara una variable en tipo numero entero.

EJEMPLO:

1 INT NUMERO = 10;

## DOUBLE

Instrucción que declara una variable en tipo numero decimal.

EJEMPLO:

1 DOUBLE NUMERO = 5.45;

## STRING

Instrucción que declara una variable en tipo de cadena de caracteres

EJEMPLO:

1 STRING CADENA = “ABDCDE”;

# Iteraciones

## FOR

Instrucción en la que se repite el mismo proceso una y otra vez dependiendo el numero de veces, usamos **TO** para definir el valor exacto del conteo del ciclo, pero si se quiere aumentar el conteo del ciclo usamos la **STEP** pero no siempre es necesario usar el STEP en FOR y para terminar el con todo el ciclo FOR usamos **NEXT**.

EJEMPLO:

1 FOR N=2 TO 0 STEP 0.25

2 PRINT “ ”; N

3 NEXT

SALIDA: 2 1.75 1.5 1.25 1 0.75 0.5 0.25 0

## WHILE

Instrucción que repite el proceso hasta que termine en **WEND** mientras trata de que se cumpla la condición.

EJEMPLO:

1 d = 0

2 WHILE d <= 10

3 PRINT “ ”; d

4 d = d+ 1

5 WEND

SALIDA: 1 2 3 4 5 6 7 8 9 10

## REPEAT

Instrucción parecida al WHILE la diferencia es que ejecuta el proceso antes de determinar la condición. Sale del ciclo cuando cumple con la condición y entonces termina con la instrucción **UNTIL**.

EJEMPLO:

1 d = 0

2 REPEAT

3 PRINT “ ”; d

4 d = d+ 1

5 UNTIL d <= 10

SALIDA: 1 2 3 4 5 6 7 8 9 10

# Funciones

## DEF

Define instrucciones(función) de usuario con valores que se pueden devolver y otros valores que no se devuelven. Fuera del DEF se pueden usar otras instrucciones como **CALL** que es la instrucción definida por el usuario para llamar a una función definida con el nombre especificado(String), ahora para devolver las instrucciones de un DEF usamos **RETURN**. Si un caso se requiere múltiples valores o salidas en un DEF se usa la instrucción **OUT** y para terminar con la definición de DEF para una función o instrucción de usuario usamos **END.**

EJEMPLOS:

1 A = CALL (“FACTORIAL”, N)

2

3 DEF FACTORIAL(N)

4 IF N == 1 THEN

5 RETURN N

6 ENDIF

7 RETURN N \* FACTORIAL(N-1)

8 END

# Metodos

## SWAP

Instrucción que cambia los valores de 2 variables.

EJEMPLO:

1 SWAP a = 23, b = 47

2 VAR temp

3 PRINT “Antes a ”; a

4 PRINT “Antes b ”; b

5 temp = a

6 a = b

7 b = temp

8 PRINT “Después a ”; a

9 PRINT “Después b ”; b

SALIDA:

Antes a 23

Antes b 47

Después a 47

Después b 23

## INC

Instrucción que incrementa el valor de una variable en +1(se agregará el valor de la expresión).

## DEC

Instrucción que disminuye el valor de una variable en -1(se restara el valor de la expresión).

EJEMPLOS:

1 VAR i = 1

2 WHILE i < 10

3 PRINT i

4 INC i

5 WEND

1 VAR i = 1

2 WHILE i > 10

3 PRINT i

4 DEC i

5 WEND

## CONTINUE

Instrucción que fuerza un ciclo para proceder se puede en FOR, WHILE Y REPEAT.

EJEMPLO:

1. VAR conteo = 10
2. WHILE conteo >= 0
3. IF conteo == 7
4. DEC conteo
5. CONTINUE
6. ENDIF
7. PRINT “ ”; conteo
8. DEC conteo
9. WEND

SALIDA: 10 9 8 6 5 4 3 2 1 0

# CONCLUSION

Se puede decir que es uno de los lenguajes mas factibles para todo aquel que quiere programar y no sabe mucho como hacerlo. Hasta por el nombre uno puede empezar por lo básico como por ejemplo programar cálculos, contar números imprimir nombres con un determinado número de espacios, etc.