Informe del Proyecto

Lenguaje Propuesto: PugCodigo

Piero Omar De La Cruz Mancilla

Universidad La Salle
Facultad de Ingenierías
Departamento de Ingeniería y Matemáticas
Carrera Profesional de Ingeniería de Software
Compiladores

2024 - A

Informe del Proyecto Final

Tema: PugCodigo

Nota

Estudiante	Escuela	Asignatura
Piero Omar De La Cruz Mancilla pdelacruzm@ulasalle.edu.pe	Carrera Profesional de Ingeniería de Software	Compiladores Semestre: V

Semestre académico	Fecha de inicio	Fecha de entrega
2024 - A	Del 23 de Junio 2024	Al 02 de Julio 2024

1. Introducción

- Esta introducción presenta la motivación y justificación del lenguaje de programación PugCodigo, un lenguaje que combina la sintaxis de los lenguajes populares con la novedad de utilizar "pugçomo punto de entrada y çharlie"para las declaraciones de funciones.

1.1. Motivación del Lenguaje PugCodigo:

- **E**l lenguaje de programación PugCodigo se crea con la finalidad de ofrecer una alternativa innovadora y amigable para el aprendizaje y desarrollo de programas en el ámbito de la ingeniería de software. Inspirado en lenguajes populares como C++ y Python, PugCodigo introduce una forma más intuitiva y accesible para que tanto principiantes como expertos puedan escribir y entender código de manera más clara y eficiente.

1.2. Características Clave:

- **Simplicidad y Claridad:** PugCodigo está diseñado para ser fácil de leer y escribir. La estructura del lenguaje permite una comprensión rápida del flujo del programa, reduciendo la complejidad que suele encontrarse en otros lenguajes más tradicionales.
- **Integración de Conceptos Familiares:** Utiliza terminología familiar como "pug"para indicar el punto de entrada principal (similar a 'main' en C++) y çharlie"para la declaración de funciones, haciendo que el código sea más accesible y memorable para los usuarios.
- **Enfoque en la Enseñanza:** Con PugCodigo, los estudiantes pueden aprender los conceptos fundamentales de la programación de una manera más lúdica y atractiva. Al reducir la barrera de entrada, los alumnos pueden centrarse en comprender la lógica y estructura del código sin sentirse abrumados por la sintaxis compleja.

2. Especificación léxica

- **Operadores de Comparacion y Aritmeticos:** +, -, *, /, ==, <=, >=, <>, <, >, #y, #o
- **Comentarios:** Los comentarios se definen asi ~texto~
- **Palabras Reservadas:** si,sino, bool, wentero, flotante,print,break,while,devuelve,caract,texto
- **Identificadores:** Los identificadores tienen que comenzar con una letra y no pueden empezar con números o subguión
- **Identificadores:** Los identificadores tienen que comenzar con una letra y no pueden empezar con números o subguión

2.1. Expresiones regulares:

Tokens	**Expresión Regular**
Comentarios	\ ∼ (\w |\W)*\∼
Identificadores	[a-zA-Z]+[\w]*
numero	0|[1-9][0-9]*
decimal	(0|[1-9][0-9]*)\.[0-9]*
bool	true|false
entero	"*entero*"
texto	"*texto*"
decimal	"*decimal*"
bool	"*bool*"
caract	"*caract*"
charlie	"*charlie*"
si	"*si*"
while	"*while*"
true	"*true*"
false	"*false*"
devuelve	"*devuelve*"
sino	"*sino*"
+	" + "
-	" − "
*	"*"
#y	"*#y*"
#o	"*#o*"
/	"/"
>	">"
<	"<"
(	"("
)	")"
[	"["
]	"]"
,	","
.	"."
=	" = "
==	" == "
<=	"<="
>=	">="
<>	"<>"
%	" %"

3. Gramática

```
Inicio->E FuncionPrincipal K
Inicio->FuncionPrincipal
E->DeclaracionFuncion E'
E'->DeclaracionFuncion E'
E'->''
DeclaracionFuncion->charlie identificador pizquierdo Parametros pderecho lizquierdo CuerpoF lderecho
FuncionPrincipal->pug pizquierdo pderecho lizquierdo Cuerpo lderecho
K->''
K->E
Parametros->''
Parametros->Y Y'
Y'->coma Y Y'
Y'->''
Y->TipoDato identificador C
TipoDato->bool
TipoDato->entero
TipoDato->flotante
TipoDato->texto
TipoDato->caract
C->cizquierdo cderecho
C->''
CuerpoF->Cuerpo J
J->''
J->devuelve Expresion
Cuerpo->''
Cuerpo->DeclaracionVariable D'
Cuerpo->Sentencias D'
D'->DeclaracionVariable D'
D'->Sentencias D'
D'->''
DeclaracionVariables->DeclaracionVariable Dc'
Dc'->DeclaracionVariable Dc'
Dc'->''
Sentencias->lizquierdo MuchasSentencias lderecho
Sentencias->si pizquierdo Expresion pderecho Sentencias sino Sentencias
Sentencias->while pizquierdo Expresion pderecho lizquierdo MuchasSentenciasWhile lderecho
Sentencias->print pizquierdo Expresion pderecho
Sentencias->identificador OPS
OPS->igual Expresion
OPS->cizquierdo Expresion cderecho igual Expresion
OPS->pizquierdo ParaLLamados pderecho
MuchasSentenciasWhile->SentenciasWhile M'
M'->SentenciasWhile M'
M'->''
MuchasSentenciasWhile->''
SentenciasWhile->lizquierdo MuchasSentenciasWhile lderecho
SentenciasWhile->si pizquierdo Expresion pderecho SentenciasWhile sino SentenciasWhile
SentenciasWhile->while pizquierdo Expresion pderecho lizquierdo MuchasSentenciasWhile lderecho
SentenciasWhile->print pizquierdo Expresion pderecho
SentenciasWhile->identificador OPS
```

```
SentenciasWhile->break
MuchasSentencias->Sentencias S'
MuchasSentencias->DeclaracionVariable S'
S'->DeclaracionVariable S'
S'->Sentencias S'
S'->''
DeclaracionVariable->TipoDato identificador Corchetes OPI
Corchetes->cizquierdo Expresion cderecho
Corchetes->''
OPI->igual Expresion
OPI->''
Expresion->Ex
Ex->Tx Ex'
Ex'->Simbolo Tx Ex'
Ex'->''
Tx->Literal
Literal->lbool
Literal->numero
Literal->ltexto
Literal->decimal
Literal->lcaract
Tx->identificador Op
Op->''
Op->cizquierdo Expresion cderecho
Op->pizquierdo ParaLLamados pderecho
ParaLLamados->''
ParaLLamados->PL PL'
PL'->coma PL PL'
PL'->''
PL->Expresion
Tx->true
Tx->false
Tx->pizquierdo Expresion pderecho
Simbolo->mas
Simbolo->menos
Simbolo->por
Simbolo->o
Simbolo->y
Simbolo->mayor_igual
Simbolo->diferente
Simbolo->menor_igual
Simbolo->comparacion
Simbolo->menor
Simbolo->mayor
Simbolo->modulo
Simbolo->dividir
Simbolo->igual
```

4. Link de github

4.0.1. https://github.com/Pdelacruz123/Compiladores.git

5. Conclusiones

Desde el análisis léxico hasta la generación de código, se logró construir un compilador funcional que traduce correctamente programas escritos en un lenguaje específico a código ejecutable. Se alcanzaron los objetivos establecidos inicialmente, como la correcta interpretación de la sintaxis del lenguaje y la generación de código eficiente.Enfrentarse a desafíos como la gestión de la tabla de símbolos, la detección de errores semánticos y la optimización del código permitió mejorar las habilidades de resolución de problemas y programación