Orlando Batres Chinchilla - 201212734
Compiladores 2

MANUAL TÉCNICO

AUGUS

TABLE OF CONTENTS

Contenido

INTRODUCCIÓN

Introducción

De la necesidad de un IDE que permitiera manejar un entorno de programación en el cual se pudiera manejar un lenguaje de alto nivel nació MINOR C IDE, un ambiente de desarrollo capaz de interpretar el lenguaje MINOR C el cual esta basado en C, el cual permite realizar acciones básicas, tales como operaciones aritméticas, lógicas y relacionales, manejo de arreglos, pilas, structs y etiquetas.

Objetivos del sistema AUGUS IDE

- Traducir el lenguaje de alto nivel MINOR C
- Manejar de manera centralizada el código que se desarrolle dentro de esta plataforma
- Proveer al usuario un ambiente de fácil usabilidad en cuanto a la interfaz grafica

Contenido técnico

ESPECIFICACIONES DE DESARROLLO

El proyecto MINOR C IDE contiene un licenciamiento tipo GNU General Public License v3.0, el cual permite utilizar el código fuente de manera comercial, modificación, distribución, uso de patentes y de uso privado.

Para tener una descripción mas detallada de lo que comprende esta licencia, visitar:

https://github.com/obatres/MinorC/blob/master/LICENSE

el código fuente puede ser encontrado en el siguiente link:

https://github.com/obatres/MinorC

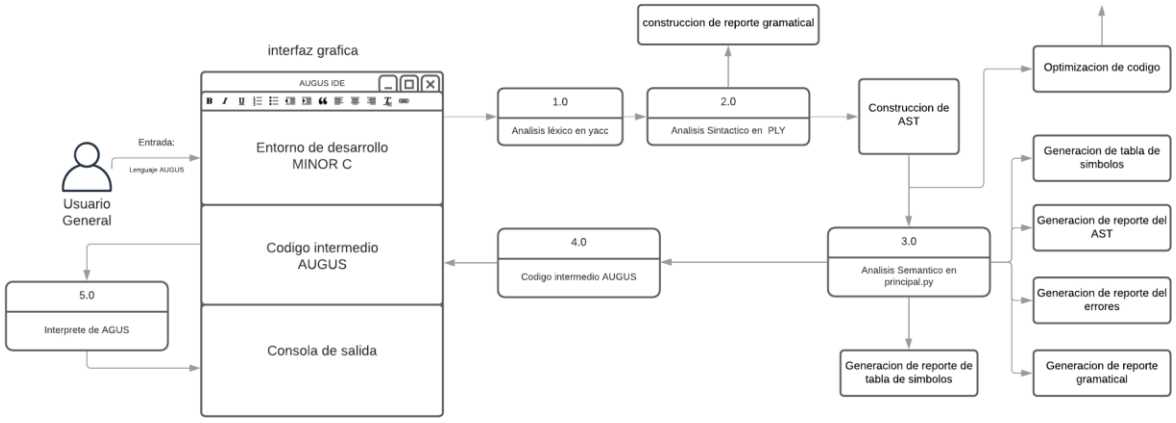
el código fuente para la interpretación de AUGUS lo puede encontrar en el siguiente link:

https://github.com/obatres/InterpretePython

HERRAMIENTAS

- PLY: Generador de analizadores léxicos y sintácticos.
- Python 3.7: Es un lenguaje de programación
- Windows 8.1: Sistema operativo
- Visual Studio Code: Editor
- PyQt5: como gestor de interfaz grafica

FLUJO DE TRABAJO



CONTENIDO TÉCNICO

LENGUAJE

Ejemplo de un archivo de entrada

```
int arreglo[10] = { 658, 245, 654, 956, 5, 754, 100, 89,

int arregloLength = 10;

void burbuja10()
{
    int i,j,aux;
    for(i = 0; i < arregloLength - 1; i++ )
    {
        for(  j = 0; j < arregloLength - i - 1; j++ )
        {
            if(arreglo[ j + 1 ] < arreglo[ j ] )
            {
                aux = arreglo[ j + 1 ];
                arreglo[ j + 1 ] = arreglo[ j ];
                                    arreglo[ j ] = aux;
            }
        }
    }
}

void imprimirBurbuja10()
{
    for( int i  = 0; i < arregloLength; i++)
    {
        printf("posicion %d: %i\n", i, arreglo[i]);
        printf("\n");
    }
}

void imprimirBurbuja11()
{
    for( int i  = 0; i < arregloLength; i++)
    {
        printf("posicion %d: %i\n", i, arreglo[i]);
        printf("\n");
    }
}

int main()
{
    printf("Arreglo Desordenado\n");
    imprimirBurbuja10();
    printf("--------------------------\n");
    burbuja10();
    printf("Arreglo Desordenado\n");
    imprimirBurbuja11();
}
```

DEFINICIONES DIRIGIDAS POR LA SINTAXIS

GRAMATICA ASCENDENTE

Terminal		Producción
INIT	→	L_INSTRUCCIONES
L_INSTRUCCIONES	→	L_INSTRUCCIONES INSTRUCCION | INSTRUCCIÓN
INSTRUCCION	→	DEFINICION_STRUCT | DECLARACION_STRUCT | DECLARACION | ASIGNACION | FUNCION
DEFINICION_STRUCT	→	"struct" Identificador "{" L_DECLARACIONES "}" ";"
L_DECLARACIONES	→	L_DECLARACIONES DECLARACION | DECLARACION
DECLARACION_STRUCT	→	"struct" Identificador LISTA_VARIABLE ";"
DECLARACION	→	TIPO L_VARIABLES ";" | TIPO L_VARIABLES "=" EXPRESION ";"
ASIGNACION	→	OPCION_ASIGNACION ";"
OPCION_ASIGN	→	L_VARIABLES "=" EXPRESION | L_VARIABLES "+=" EXPRESION | L_VARIABLES "-=" EXPRESION | L_VARIABLES "*=" EXPRESION | L_VARIABLES "/=" EXPRESION | L_VARIABLES "%=" EXPRESION | L_VARIABLES "<<=" EXPRESION | L_VARIABLES ">>=" EXPRESION | L_VARIABLES "&=" EXPRESION | L_VARIABLES "^=" EXPRESION | L_VARIABLES "|=" EXPRESION
FUNCION	→	TIPO Identificador "(" L_PFORMALES ")" BLOQUE
L_PFORMALES	→	L_PFORMALES "," PFORMAL | PFORMAL
PFORMAL	→	TIPO Identificador

DEFINICIONES DIRIGIDAS POR LA SINTAXIS

TIPO		"int" | "char" | "double" | "float" | "void"
L_VARIABLES	→	L_VARIABLES "," VARIABLE | VARIABLE
VARIABLE	→	Identificador | Identificador L_INDICES
L_INDICES	→	L_INDICES INDICE | INDICE
INDICE	→	"[" "]" | "[" EXPRESION "]"
BLOQUE	→	"{" L_SENTENCIAS "}" | L_SENTENCIAS
L_SENTENCIAS	→	L_SENTENCIAS SENTENCIA | SENTENCIA
SENTENCIA	→	DECLARACION | ASIGNACION | LABEL | GOTO | IF | SWITCH | WHILE | DO-WHILE | FOR | BREAK | CONTINUE | RETURN | LLAMADA ";"
LABEL		Identificador ":"
GOTO		"goto" Identificador ";"
IF	→	INST_IF | INST_IF L_INST_ELSEIF | INST_IF L_INST_ELSEIF INST_ELSE | INST_IF INST_ELSE
INST_IF	→	"if" "(" EXPRESION ")" BLOQUE
L_INST_ELSEIF	→	L_INST_ELSEIF INST_ELSEIF | INST_ELSEIF
INST_ELSEIF	→	"else" "if" "(" EXPRESION ")" BLOQUE

DEFINICIONES DIRIGIDAS POR LA SINTAXIS

INST_ELSE	→	"else" BLOQUE
SWITH	→	"switch" "(" EXPRESION ")" "{" DEFAULT "}" | "switch" "(" EXPRESION ")" "{" L_CASE "}" | "switch" "(" EXPRESION ")" "{" L_CASE DEFAULT "}"
L_CASE	→	L_CASE CASE | CASE
CASE	→	L_EXP_CASOS BLOQUE
L_EXP_CASOS	→	L_EXP_CASOS "case" EXPRESION ":" | "case" EXPRESION ":"
DEFAULT	→	"default" ":" BLOQUE
WHILE	→	"while" "(" EXPRESION ")" BLOQUE
DOWHILE	→	"do" BLOQUE "while" "(" EXPRESION ")" ";"
FOR	→	"for" "(" EXPRESION ";" EXPRESION ";" EXPRESION ")" BLOQUE
BREAK	→	"break" ";"
CONTINUE	→	"continue" ";"
RETURN	→	"return" ";" | "return" EXPRESION ";"
EXPRESION	→	ARITMETICO | RELACIONAL | LOGICO | BITBIT | TERNARIO | SIZEOF | CAST | INC_DEC | PUNTERO | LLAMADA | ACCESO | PRIMITIVO
ARITMETICO		EXPRESION "+" EXPRESION | EXPRESION "-" EXPRESION | EXPRESION "*" EXPRESION | EXPRESION "/" EXPRESION | EXPRESION "%" EXPRESION | "-" EXPRESION
RELACIONAL		EXPRESION "==" EXPRESION | EXPRESION "!=" EXPRESION | EXPRESION "<" EXPRESION | EXPRESION ">" EXPRESION | EXPRESION "<=" EXPRESION | EXPRESION ">=" EXPRESION

DEFINICIONES DIRIGIDAS POR LA SINTAXIS

LOGICO	EXPRESION "&&" EXPRESION | EXPRESION "||" EXPRESION | "!" EXPRESION
BITBIT	EXPRESION "<<" EXPRESION | EXPRESION ">>" EXPRESION | EXPRESION "&" EXPRESION | EXPRESION "|" EXPRESION | EXPRESION "^" EXPRESION | "~" EXPRESION
TERNARIO	EXPRESION "?" EXPRESION ":" EXPRESION
SIZEOF	"Sizeof" "(" EXPRESION ")"
CAST	"(" TIPO ")" EXPRESION
INC_DEC	"++" Identificador | "--" Identificador | Identificador "++" | Identificador "--"
PUNTERO	"*" Identificador | "**" Identificador | "&" Identificador
LLAMADA	Identificador "(" PRMS_ACT ")"
PRMS_ACT	PRMS_ACT "," EXPRESION | EXPRESION
ACCESO	Identificador "." LISTA_ACCESO
LISTA_ACCESO	LISTA_ACCESO "." Identificador | Identificador
PRIMITIVO	Entero | Caracter | Doble | Flotante | Cadena | Identificador