**Proyecto de compiladores**

**Otoño 2018**

**Dra. Hilda Castillo Zacatelco**

Escribir un programa intérprete que reciba como entrada el código que más adelante se detallará y dé como salida la interpretación de ese código.

El intérprete deberá realizar el análisis léxico, sintáctico y semántico del programa fuente, además debe interpretar el código de entrada a través de un ambiente gráfico

El código de entrada contiene sentencias involucradas con un ciclo para, con si, con operaciones relacionales y de igualdad, con asignación, operaciones aritméticas y definición y uso de arreglo de datos. Los arreglos solo pueden ser de tipo entero sin signo y carácter. Las variables serán de tipo entero y carácter, y no será necesario definirlas. Se podrá hacer uso de constantes enteras y caracter.

La escritura del programa no será sensible a mayúsculas y minúsculas, es decir, la variable cont es la misma que *Cont* y que CONT. La única excepción es con el uso de constantes, esto es, ‘a’ es diferente de ‘A’.

Las instrucciones pueden ir seguidas o una por renglón, puede haber espacios y/o tabuladores antes o después de las instrucciones o entre tokens.

**Estructura general de un programa:**

**PROGRAMA** *Nombre\_programa*

*Definición e inicialización de constantes*

*Definición e inicialización de arreglos*

**INICIO**

*Instrucciones*

**FIN**

Donde:

*Nombre\_programa* es un identificador que será el nombre del programa, éste deberá está formado por una combinación de letras, dígitos y ‘\_’ (guión bajo), donde el primer carácter es una letra o ‘\_’. La longitud de la cadena deberá ser al menos de 1.

**Operadores**

**Operadores aritméticos**: +, -, \*, / (división entera, devuelve cociente), MOD (devuelve residuo)

**Operadores relacionales**: <, >, <=, >=

**Operadores de igualdad:** == (igual a ), != (diferente)

***Definición e inicialización de constantes***

En esta sección se podrán definir constantes enteras y de tipo carácter. Si se coloca la palabra reservada CONSTANTES entonces se debe definir al menos una constante. No se podrá definir una constante para dos valores diferentes. La sintaxis es la siguiente:

**CONSTANTES**

Nombre\_variable1= valor1

.

.

Nombre\_variablen= valorn

Donde Nombre\_variablei es un identificador y valori es el valor que se le asocia a la constante que puede ser entero o carácter.

Por ejemplo:

CONSTANTES

MAX=67

CAR=’B’

***Definición e inicialización de arreglos***

En esta zona se podrán definir e iniciar uno o más arreglos unidimensionales. Si se coloca la palabra ARREGLOS se debe definir al menos un arreglo. Todos los arreglos tendrán asociada una variable LENGTH que serán el número de elementos del arreglo y se invocará por medio del nombre del arreglo seguido de ‘.’ y la palabra LENGTH (por ejemplo arreglo.length). Tómese en cuenta que el primer dato de un arreglo estará en la posición 0. La siguiente sintaxis es la siguiente:

**ARREGLOS**

Nombre\_arreglo **=** **{** dato0**,** dato1**,** dato2**,** ………**,** daton}

Los datos sólo pueden ser enteros sin signo o caracteres encerrados entre comillas simples. El intérprete detectará el tipo de los argumentos (enteros o caracteres). Si se colocan valores mezclados entonces se generará un error. Lo mismo ocurrirá si se redefine la misma variable. Para accesar a los datos de un arreglo se debe usar la siguiente sintaxis:

Nombre\_arreglo[posición]

Ejemplo:

Datos = { 12, 66, 77, 12, 7 888, 1113, 56}

Valores={9,0,76,54,12}

varios1= {‘k’, ‘8’, ‘3’}

arrConError = {‘m’, ‘h’, 8, ‘fg’} Esto generaría un error.

***Instrucciones***

Son las instrucciones que se interpretarán y pueden ser siguientes:

**Ciclo *PARA***

La semántica de esta instrucción es la misma que para el for de lenguaje C.

Sintaxis:

**PARA** variable\_control = limite1 o variable **HASTA** limite2 o variable **PASO** incremento o decremento **HACER**

*Instrucciones*

**FIN**

*variable\_control* es la variable de control para el ciclo PARA, que se incrementará al finalizar cada iteración de acuerdo con *incremento* o *decremento.*

*limite1* es el valor entero que inicialmente tomará variable\_control.

*limite2* es el valor entero que tomará variable\_control en la última iteración.

*limite1* y *limite2* son valores enteros sin signo.

*incremento* es el valor con el cual se incrementará variable\_control en cada iteración, se le antepondrá el signo más ‘+’.

*decremento* es el valor con el cual se decrementará variable\_control en cada iteración, se le antepondrá el signo menos ‘-‘.

Ejemplo:

PARA i=1 HASTA 20 PASO +1 HACER

X=X+2;

A[i]=A[i+2]+3;

FIN

**Condicional SI**

Tendrá la misma semántica que un if en lenguaje de programación C.

Sintaxis:

**SI** **(***condición***)** **ENTONCES**

*Instrucciones*

**SINO**

*Instrucciones*

**FIN**

**O**

**SI (condicion) ENTONCES**

*Instrucciones*

**FIN**

*condición* es una proposición simple que retorna verdadero o falso. Su sintaxis es la siguiente:

( variable o constante símbolo\_relacional o de igualdad variable o constante)

Ejemplos:

(abc < 45)

(car == ‘a’)

(45 >= cont)

**Escribe**

Con esta instrucción se podrán escribir variables o constantes, pero solo acepta un argumento. Sintaxis:

ESCRIBE (variable o constante);

***Lee***

Esta instrucción permite leer un valor desde teclado, solo acepta un argumento el cual debe ser de tipo entero o de tipo carácter. La variable deberá estar inicializada antes de realizar la lectura.

Sintaxis:

LEE(car);

car debió haberse iniciado antes para que el intérprete conozca de qué tipo es.

***Expresiones***

Las expresiones, en este lenguaje, serán expresiones simples de a lo más dos operandos y un operador, siempre asociadas a una asignación . Sintaxis:

variable = variable o constante operador variable o constante;

Ejemplo:

X=par + 4;

Car=45;

Vec=’s’;

Mat=7 \* x4;

**Ejemplos de programas que deberá aceptar el intérprete:**

PROGRAMA EJEMPLO

CONSTANTES

NUM=23

CAR=’X’

ARREGLOS

DATOS={3,2,1,0}

INICIO

ESCRIBE(‘A’);

ESCRIBE(‘=’);

SI (DATOS.LENGTH >1) ENTONCES

PARA var=1 HASTA datos.LENGTH PASO +1 HACER

ESCRIBE(DATOS[i]);

FIN

FIN

FIN

Programa burbuja

Arreglos A={5, 8, 2, 9, 1}

Inicio

Para i=2 hasta a.length paso +1 hacer

P= a.length-i;

Para j=0 hasta p paso +1 hacer

Si (a[j] > a[j+1]) entonces

Aux=a[j]; A[j]=a[j+1]; A[j+1]=aux;

fin

fin

fin

fin

programa pares

constantes num=5

arreglos valores={5, 7, 2, 4, 8}

inicio

car=’1’;

lee(car);

lim=valores.length-1;

si (car==’p’) entonces

para var=lim hasta 0 paso -1 hacer

par=valores[var] mod 2;

si (par!=0) entonces

valores[var]=valores[var]+1;

fin

fin

sino

si (car==’i’) entonces

para var=lim hasta 0 paso -1 hacer

impar=valores[var] mod 2;

si (par==0) entonces

valores[var]=valores[var]+1;

fin

fin

sino

Escribe(‘0’);

fin

fin

fin