Instituto Tecnológico de Estudios Superiores de Monterrey

Diseño de compiladores

Documentación del Compilador para el Lenguaje “Zero Two”

Alumno:

Ivan Eloy Bonilla Gameros A00826077

Introducción:

El compilador para el Lenguaje “Zero Two” es una herramienta diseñada para facilitar el aprendizaje de los fundamentos de la programación en general. Este lenguaje está especialmente orientado a jóvenes interesados en explorar el mundo de la programación.

Estructura general

Para comenzar a programar es necesario que definamos un nombre del programa que vamos a crear. Para esto podemos hacer uso de la palabra reservada ‘Program’ seguido del nombre que deseamos utilizar para este programa. Después viene el bloque de ‘Variables’ donde podemos definir todas nuestras variables globales del programa.

Además de esto es importante agregar el bloque principal con la palabra reservada ‘main’ y esta se encargará de guiar nuestro programa por completo.

```
Program Hiro;

Variables {
  string msg;
  msg = “Hola”;
}

main() {
  print: msg;
}
```

Declaración de variables

La declaración de variables es un paso muy importante en los lenguajes de programación, y difiere entre estos. Para este caso se esta utilizando una sintaxis simple para facilitar la comprensión, se pueden definir variables de tipo int, float, bool y string. Y es muy importante que como este es un programa con el propósito de ser sencillo para aprender a programar, se tiene que seguir un orden estricto, esto es que tienes que declarar primero la variable para luego poderle asignar un valor, esto no se puede hacer en la misma línea e igualmente se declaran variables una por una.

```
Program example;

Variables {
  int x;
  float y;
  bool t;
  string msg;
}
```

```
main() {
    x = 1;
    y = 2.5;
    t = false;
    msg = "Hello, world!";
}
```

Uso de operadores

Este lenguaje soporta el uso de operadores relacionales así como también aritméticos. Es muy importante tomar en cuenta que para que funcionamiento correcto del programa ocupas utilizar solo los operadores relacionales siguientes: <, <=, >, >=, ==, != y aritméticos: +, -, *, /.

```
Program example;

Variables {
  int x;
  int y;
}

main() {
    x = 1;
    y = 2;
    int r;
    r = x + y;
    bool b;
    b = 2 < 3;
}
```

Estatutos Condicionales

Una vez que hayas practicado con estos operadores es importante aprender a usar los estatutos condicionales, if o if-else. Para una compilación correcta de estos es necesario que después de usarlos utilices un “;”, como en el ejemplo:

```
Program example;

Variables {
  int x;
  int t;
  bool z;
```

```
}

main() {
  x = 5;
  t = 2;
  int f;

  if (x > 2) {
      print: x;
  };
}
```

Otro ejemplo con if-else seria:

```
Program example;

Variables {
  int x;
  int t;
  bool z;
}

main() {
  x = 5;
  t = 2;
  int f;

  if (x > 2) {
      print: x;
  } else {
    print: t;
  };
}
```

Estatuto “print”

Como vimos en el ejemplo anterior tenemos un estatuto nuevo, este es la operación “print”, de nuevo aquí la sintaxis es muy importante, ya que siempre que queramos imprimir algo la declaración correcta de este estatuto es “print: variable;”

Ciclos

Los ciclos son una parte muy importante de nuestro lenguaje de programación, por ahora solo tenemos el estatuto “while” implementado. Este tiene una sintaxis similar al if, recuerda que al final se tiene que usar un semicolon como en este ejemplo:

```
Program example;

Variables {
  int x;
  int t;
  bool z;
}

int fact(int a) {
  int y;
  y = 1;
  int result;
  result = a;
  while (y < a) {
    result = result * y;
    y = y + 1;
  };

  return result;
}

main() {
  x = 5;
  t = 2;
  int f;

  f = fact(x);
  print: f;
}
```

Declaración y llamada de funciones

Como en el ejemplo anterior también tenemos la habilidad de declarar funciones y llamarlas en nuestro bloque main.

Para la declaración de una variable es muy importante la sintaxis de primero poner el tipo de retorno que tendremos. Como en el ejemplo pasado el tipo es ‘int’ y la variable que vayamos a regresar también tiene que ser de ese tipo. El compilador se encarga de hacer esta comparación antes, si no te arrojara un

error. Entonces ponemos el tipo, después el nombre de la función y dentro de los paréntesis los parámetros que necesitara la función.

Esta sintaxis es muy importante para que funcione correctamente nuestro compilador, igualmente para llamar a la función, la variable que recibirá el retorno de la función ocupa ser el mismo tipo, como en el ejemplo la variable 'f' es 'int'.

Funciones prederteminadas para este proyecto

Como la intención de este proyecto, era tratar de enseñar a personas que no saben programar, se preparó unas funciones que podemos llamar para que nos de una idea simple de como hacer ciertas acciones.

```
Program example;

Variables {

}

main() {
ayuda();
}
```

Esta función ayuda nos genera esto:

Estas son todas las funciones de 'enseñanza' que puedes usar:

- teach_sum
- teach_substraction
- teach_multiplication
- teach_division
- teach_if
- teach_while
- teach_function_declaration
- teach_function_call

Que si intentamos llamar una de estas, de la misma manera, nos genera esto:

```
Program example;
```

```
Variables {

}

main() {
teach_while();
}
```

Un bucle while ejecuta repetidamente un bloque de código mientras una condición sea verdadera. La sintaxis es 'while (condición) { // código a ejecutar mientras la condición es verdadera }'.

Por el momento es muy simple, pero se busca en una versión futura aumenta la descripción y que en lugar de solo una descripción, esta sea interactiva y te vaya guiando paso a paso.