Universidad Tecnológica Centroamericana

Compiladores 2

Catedrático: Ing. Carlos Vallejo

Manual

Estudiantes:

Andrea Escobar - 11641363

Eduardo Guevara - 11711078

Michelle Lopez - 11641402

Tegucigalpa, M. D. C.

22 de diciembre del 2020

Lenguaje: WAP

1. Identificadores:

a. Todos los identificadores de cualquier tipo comienzan con underscore(_).
b. Utilizamos tipado estático
c. Ejemplos:
    i. int _variable
    ii. char _variableDeChar
    iii. bool _valorBooleano
    iv. func _nombreDeFuncion

2. Estructura de las funciones:

a. Todas las funciones deben ir antes del main.
b. Todas las funciones comienzan con la palabra reservada “func”.
c. Después de la palabra reservada func va el tipo de valor de retorno.
d. En tercer lugar va el identificador de la función.
e. Por último va los paréntesis, y dentro de los paréntesis van los parámetros que la función recibe.
f. Los parámetros dentro del paréntesis en caso de ser varios van separados por coma.
g. Ejemplo:
    i.
```
func int _funcion () begin
        return 1;
end func
```
    ii.
```
func bool _funcionBooleana (bool _t) begin
        return _t;
end func
```

3. Bloque Condicional (IF):

a. Todos los bloques condicionales comienzan con la palabra reservada “if”, seguido por la condición entre paréntesis.
b. Ejemplo:
    i.
```
if (_a > _b) begin
        print (“A es mayor que B”);
end if
```

4. Bloque Condicional (ELSE):

a. Todos los bloques condicionales del tipo else comienzan con la palabra reservada “else”.
b. Ejemplo:
    i.
```
else begin
        print (“B es Mayor que A”);
    end else
```

5. Estructura de Ciclo (FOR):

a. Todos los ciclos del tipo for comienzan con la palabra reservada “for”, seguido de la palabra reservada “int”, luego una variable de control para el ciclo y un operador hasta “->” para indicar la cantidad de veces que este va a realizar el ciclo, luego de eso va la variable seguida de un “++” que significa un incremento de uno en uno, todo esto entre paréntesis.
b. El operador hasta es inclusivo, es decir, la parte va desde la parte izquierda (tomándolo en cuenta, siempre es 0) hasta el lado derecho (tomando en cuenta).
c. Ejemplo:
    i.
```
for (int _i -> 2; _i++) begin
        print (_i);
    end for
```

6. Ciclo con Condición (Loop):

a. Todos los ciclos con condición comienzan con la palabra reservada “loop” y terminan con un end loop.
b. Ejemplo:
    i.
```
loop (_boolean & (true == true)) begin

        _boolean = false;

        print(_matriz[0,0]);

    end loop
```

7. Estructura Multi Condición (Options):

a. Todas estas estructuras comienzan con la palabra reservada “options”, seguido de una variable a evaluar y el tipo de esa variable.
b. Cada opción de esta estructura es denominada “opt” y a esta palabra la siguen corchetes encerrando el valor a comparar.
c. Toda instrucción Options debe tener una opción por defecto, denominada opt_def.

d. Ejemplo:
    i. options (_i , int) begin

```
opt [1]{

    _i = 2;

    print(true);

}

opt_def {

    _j = 3;

}
```

end options

8. Salida y Entrada de Datos:

a. Para la salida de datos se utiliza la instrucción “print”.
b. Para la entrada de datos se utiliza la instrucción “read”.
c. Ejemplos:
    i. print (“Hola”);
    ii. print(_variable);
    iii. read _i;
    iv. read _variable;

9. MAIN:

a. El main lleva al igual que una función normal, empieza con la palabra reservada “func”.
b. Hace uso de la palabra reservada “int_main()”.
c. Ejemplo:
    i. func int_main() begin

```
...
```

end main

10. Comentarios:

a. Los comentarios se hacen por medio de /* para abrir y */ para cerrarlos.
b. Ejemplo:
    i. /* Este es un comentario */

11. Fin de Línea:

a. Todas las instrucciones simples (No estructuras) terminan con un “;”.
b. Ejemplos:

i. int _entero;
ii. _entero = 5;
iii. print (_entero);
iv. read _entero;

12. Tipos:

Tipo	Asignación	Acceder a una posición
Int	5	-
char	‘a’	-
bool	true | false	-
array_int{size}	{1,2,3}	_nombrearray[0]
array_char{size}	{‘a’,’b’,’c’}	_nombrearray[0]
array_bool{size}	{true}	_nombrearray[0]
matrix_int{size1, size2}	{{1,2}, {4, 7}}	_nombrematriz[0,0]
matrix_char{size1, size2}	{{‘a’,’b’}, {‘c’,’d’}}	_nombrematriz[0,0]
matrix_bool{size1, size2}	{{true,false},{false,false}}	_nombrematriz[0,0]

13. Operadores:

Operadores	Función
=	Asignación

==	Comparación
!=	Distinto
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
+	Suma
-	Resta
*	Multiplicación
/	División
->	Hasta (uso en for)
;	final de línea
++	sumar 1
&	AND
|	OR
!	NOT

Ejemplos de Código Fuente

Funcionales

---

func int _pruebaoptions(char _u) begin

```
    /*print("TEST\n");*/
    options (_u , char) begin
            opt ['a']{
            print("El caracter es: a\n");
            }
        opt ['b']{
            print("El caracter es: b\n");
        }
            opt_def {
            print("El caracter es: ");
            print(_u);
        }
    end options
    return 5;
end func
func int_main() begin
    int _c = 5;
    loop (_c > 5) begin
        print(_c);
        print("\n");
        _c = _c - 1;
    end loop
    char _o = 'x';
    _c = _pruebaoptions(_o);
    print("\n");
    print(_c);
end main
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
func int _potencia(int _m, int _n) begin
    if(_n == 1) begin
        return _m;
    end if
    return _m * _potencia(_m, _n - 1);
end func
func int_main() begin
        int _a, _b;
        print("Introduzca un numero:\n");
        read _a;
        print("Introduzca otro numero:\n");
        read _b;
        print("El resultado es: ");
        print(_potencia(_a, _b));
        print("  ");
end main
-----------------------------------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------------------------------
func int _prueba(int _f, bool _h, int _g, int _o, bool _x, int _este) begin
```

```
    print("El quinto parametro es: ");
    print(_x);
    print("\n");
    print("El sexto parametro es: ");
    print(_este);
    print("\n");
end func
func int _fibonacci(int _l) begin
    if(_l == 1) begin
        return 1;
    end if
    if(_l == 2) begin
        return 1;
    end if
    return _fibonacci(_l - 1) + _fibonacci(_l - 2);
end func
func int_main() begin
    int _num;
    print("Prueba\n");
    _prueba(1,true,2,3,false,40);
    print("Ingrese un numero: ");
    read _num;
    for(int _v -> _num, _v++) begin
        print(_v);
    end for
    print("\n");
    print(_fibonacci(5));
end main
```

No-Funcionales

```
func int _pruebaoptions(char _u) begin
    /*print("TEST\n");*/
    options (_u , char) begin
            opt [2]{
            print("El caracter es: a\n");
            }
        opt [1]{
            print("El caracter es: b\n");
        }
            opt_def {
            print("El caracter es: ");
            print(_u);
        }
```

```
    end options
    return 5;
end func
func int_main() begin
    int _c = 'a';
    loop (_c > false) begin
        print(_c);
        print("\n");
        _c = _c - 1;
    end loop
    char _o = 'x';
    _c = 5 * 5 + _pruebaoptions(_c);
    print("\n");
    print(_c);
end main
```

```
func int _potencia(int _m, int _n) begin

    if(_n == 1) begin

        return _m;

    end if

    return _m * _potencia(_m, _n - 1);

end func

func int_main() begin

        int _a, _b;

        print("Introduzca un numero:\n");

        read _a;

        print("Introduzca otro numero:\n");

        read _b;

        print("El resultado es: ");

        _a = _potencia(_a, _b, 20);

        print("  ");
```

```
end main
```

```
func int _prueba(int _f, bool _h, int _g, int _o, bool _x, int _este) begin
    print("El quinto parametro es: ");
    print(_x);
    print("\n");
    print("El sexto parametro es: ");
    print(_este);
    print("\n");
end func
func int _fibonacci(int _l) begin
    if(_l == 1) begin
        return 1;
    end if
    if(_l == 2) begin
        return 1;
    end if
    return _fibonacci(_l - 1) + _fibonacci(_l - 2);
end func
func int_main() begin
    int _num;
    print("Prueba\n");
    _prueba(1,true,2,3);
    print("Ingrese un numero: ");
    read _num;
    char _caracter = 'a';
    for(int _v -> _caracter, _v++) begin
        print(_v);
    end for
    _x = 15;
    print("\n");
    print(_fibonacci(5));
end main
```

Manual Técnico

Al ejecutar el programa se debe proceder a cargar un archivo por medio del botón buscar. Una vez seleccionado el archivo se debe dar click en análisis para comenzar el proceso de compilación con las tres primeras fases: Análisis Léxico, Análisis Sintáctico y Análisis Semántico. Una vez terminadas las 3 fases de no presentarse error se debe preceder a generar los cuadruplos, apretando el boton

cuadruplos el cual los genera y los muestra en una tabla, despues de eso es necesario regresar y proceder con el último paso que es la generación de código final, para esto se debe apretar el botón final y luego se crea un archivo para guardar tal código y poder ejecutarlo luego.