

Diseño y Construcción de Compiladores

Semántica del subconjunto de la gramática de C++

1. La longitud de los identificadores no puede superar los 8 caracteres.
2. Se hace distinción entre mayúsculas y minúsculas.
3. Los comentarios en los programas comienzan y terminan con # dentro de la misma línea.

Variables

4. Las variables deben ser declaradas antes de su uso.
5. Las variables simples no pueden ser declaradas con tipo **void**.
6. Las variables tienen alcance de bloque.

Arreglos

7. El tipo base de los arreglos puede ser: **char**, **int** o **float**.
 8. La cantidad de elementos de un arreglo puede estar dada por un número natural (es decir, mayor a 0) **y/o** a través de la inicialización del mismo.
 9. Los **índices válidos** del arreglo son valores enteros entre 0 y tamaño-1.
- Cuidado, y/o, o constante o inic y si ambas que corresponda
- Constantes puedo controlar, guarda con variables

Asignación

10. Los tipos de ambos lados de la asignación deben ser estructuralmente equivalentes.
11. No se permite la asignación de arreglos como un todo.
12. La asignación es un operador que retorna el valor asignado.
13. En la inicialización de arreglos en el momento de la declaración se debe chequear que la cantidad de valores sea igual o menor a la constante y que el tipo también se corresponda.
14. En el lado izquierdo de una asignación debe haber una variable.

Coerciones

tipo	operado con	tipo	retorna
char		int	int
char		float	float
int		float	float

Entrada-Salida

15. La constante **string** sólo puede aparecer en las proposiciones de E/S.

16. La aparición de los caracteres `\n` o `\t` en una constante **string** se interpretarán como *new line* o *tab*, al igual que en C.
17. **cout** sólo baja de línea si existe un **string** con `\n`.
18. **cin** baja de línea al analizar el ingreso de todos los valores de esa proposición.
19. Las proposiciones de E/S aceptan variables y/o expresiones de tipo **char**, **int** y **float**.

Funciones-Parámetros

20. El tipo del valor de retorno de una función/procedimiento debe ser: **void**, **char**, **int** o **float** y debe coincidir con el especificador de tipo de la función/procedimiento.
21. Si el tipo de la función es **char**, **int** o **float**, la proposición **return** es obligatoria. En caso contrario, no debe haber una proposición de retorno.
22. La clase del identificador de la función/procedimiento debe ser función, es decir, no puede ser ni una variable ni un parámetro.
23. El reconocimiento de los parámetros se realiza por posición.
24. Los tipos de pasajes de parámetros son: por valor y por referencia.
25. El parámetro formal por referencia se especifica como en C++, Por ej.: **int & a**.
26. La cantidad de parámetros reales debe coincidir con la cantidad de parámetros formales.
27. El nombre de un arreglo en el parámetro real implica proveer la dirección del arreglo. En este caso el parámetro formal debe ser de nido como pasaje por valor.
28. No se permite **<tipo> & <nombre arreglo> []** en la definición de un parámetro.
29. Si el tipo de pasaje es por referencia, entonces el parámetro real debe ser una variable.
30. Si el pasaje es por valor, el parámetro real es una expresión.
31. **main()** debe ser un procedimiento sin parámetros que tiene que aparecer exactamente una vez en el programa fuente.

a(int x, float b)
si llamo a a, le
tengo que pasar
los param en
orden

Lógica Booleana

32. Los operandos de los operadores lógicos o relacionales pueden ser **char**, **int** o **float**.
33. Las condiciones de las proposiciones de selección e iteración pueden ser de tipo **char**, **int** y **float**. El valor 0 se considera **falso**, cualquier otro valor se considera **verdadero**.
34. Los operadores relacionales retornan 0 para **falso** o 1 para **verdadero**.
35. Los operadores lógicos retornan 0 para **falso** o 1 para **verdadero**.