PROCESADORES DEL LENGUAJE

ENTREGA FINAL DE LA PRÁCTICA

Bittor Alaña Olivares & Francisco Javier Blázquez Martínez

Resumen

En este documento presentamos brevemente los ligeros cambios respecto a la especificación inicial, e introducimos algunos casos de uso que ejemplifican el uso del programa.

Cambios

-Los campos de los registros se acceden a partir de ahora mediante el operador punto '.', no entre signos de desigualdad '<>'. Ejemplo:

<int,bool,int> registro; registro.0 = 0; registro.1 = true; registro.2 = -5;

-Eliminada la palabra reservada null. La integridad de los punteros, y el uso correcto de los mismos quedan a cargo del programador. Si se utiliza el operador & para conseguir una dirección de memoria, es el programador quien debe cerciorarse de que el uso de ese punto no salga del ámbito de la variable de la cual ha tomado la dirección.

-Los punteros van a apuntar únicamente a tipos básicos (int, bool). Al tomar la dirección de una variable, se guarda su dirección base. Uno puede acceder a valores consecutivos a partir de la dirección de memoria que almacena el puntero con corchetes [], siendo responsable el mismo programador de no salir de la zona reservada de memoria:

*int v; new(v,10); v[5] = 0; (Caso 1)

[int, 10] v; *int ptr = &v; v[0] = 1; (Caso 2)

Características

-El paso de argumentos se realiza por copia.

-Las funciones devuelven tipos de tamaño uno: enteros, booleanos, o un puntero a enteros o booleanos.

-Se permiten procedimientos que no devuelvan nada (void).

-Existe el manejo de memoria dinámica. Instrucción especial **new** que acepta dos parámetros: un puntero a int o bool, y una expresión aritmética, que se evalúa en *RunTime* para reservar la cantidad necesaria. La instrucción **new** se encarga de que el puntero apunte a la nueva dirección de memoria.

-La visibilidad entre funciones es total: todas las funciones están al mismo nivel y se pueden llamar las unas a las otras, salvo la función main. La palabra main es una palabra reservada para la función principal, que debe estar al final de todos los procedimientos. Las recursiones cruzadas son válidas.

-La ejecución comienza reservando espacio para el main y poniendo a cero todos los elementos de su marco de activación salvo el valor del retorno, que se coloca en la posición de la instrucción **stp** que sucede inmediatamente al código máquina del main.

Modo de ejecución

Si se ejecuta el proyecto sin pasar parámetros al Main, se ejecuta un ejemplo por defecto, mostrando su salida por pantalla. Opcionalmente se puede ejecutar con uno o dos argumentos de entrada **input** y **output**. **input** es el código fuente en nuestro lenguaje a compilar, y se escribe el código P en **output**. Si se dan ambos argumentos, se imprime el árbol de sintaxis abstracta por consola. Si solo se da uno, se imprime el código P en la consola.

Ejemplos

La carpeta **test-files** contiene una gran cantidad de ficheros de prueba. Entre ellos se cuentan varios concretos y muy especializados con el objeto de testear las funcionalidades básicas del lenguaje, y otros ejemplos de algoritmos típicos. Para probar los ficheros de

prueba básicos de generación de código de **test-files/code-gen/** se puede ejecutar el script de Bash **run-testfiles.sh**, que compara las salidas con las esperadas. Estos ficheros comprueban todas las instrucciones con bastante exhaustividad.

En la carpeta **test-files/code-gen/** hay ejemplos para probar la recursión, el uso de memoria dinámica, el paso de parámetros por copia, etc. Se puede probar el algoritmo de ordenación *quicksort* y la búsqueda binaria *busqbin*, además de haber algún ejemplo de uso de matrices.