Asterix

Impertative programming language

Sergio Domínguez Cabrera y Enrique Carro Garrido

No, Obélix. Tú no tendrás la poción mágica.
Asterix el Galo [1]

En el presente texto se realiza una descripción detallada del lenguaje de programación imperativo `Asterix` .

Introducción a Asterix

`Asterix` nace de la idea de ser un lenguaje de programación de paradigma imperativo, con tipado estático y portable. Los programas escritos en este lenguaje generan WebAssembly al compilarse pueden ser interpretados por un intérprete de este. Los programas escritos en `Asterix` tienen que tener la extensión .atx. La codificación de los ficheros .atx es UTF-8.

Tipos básicos

Por defecto, `Asterix` es capaz de distinguir 4 tipos básicos:

- Integerix: Representa el subconjunto finito de los números enteros que pueden ser expresados con 4 bytes. Para declarar una variable con este tipo hay que usar la palabra reservada intix.
- Charix: Representa el conjunto de caracteres de la codificación UTF-8. Para declarar una variable con este tipo hay que usar la palabra reservada charix.

[1] Esta es una frase famosa que se repite en toda la saga. Cuando era niño, Obélix cayó en un caldero de poción mágica, cosa que ocasionó efectos a perpetuidad. Sin embargo, eso no le impide a Obélix intentar obtener una poción mágica cada vez que se presenta la oportunidad.

- BOOLEANIX: Representa los valores de lógica binaria, esto es dos valores, que en nurestro lenguaje represetamos como galo (true) y romano (false). Empleamos un byte para almacenar estos valores. Para declarar una variable con este tipo hay que usar la palabra reservada boolix.
- ENUMERATIX: Representa un tipo ordinal cuyo orden se indica por la disposición de los valores en la definición. Para declarar una variable con este tipo hay que usar la palabra reservada enumix.
- FLOATIX: Representa el subconjunto finito de los números racionales que pueden ser expresados con 8 bytes. Para declarar una variable con este tipo hay que usar la palabra reservada floatix.

`Asterix` permite agrupar estos tipos en VECTIX que son listas ordenadas de elementos de un único tipo (que también pueden ser VECTIX. Para declarar una varible como VECTIX hay que usar la palabra reservada vectix.

`Asterix` también soporta la creacion de CALDERIX que son una agrupación de variables que pueden ser de tipos distintos (incluso pueden contener otros CALDERIX). Para declarar una variable con este tipo hay que usar la palabra reservada caldix.

Declaración de variables

Para declarar una variable hay que seguir el siguiente formato: lo primero que escribimos es el tipo del que queremos que sea, seguidamente el identificador y también tenemos la posibilidad de que tenga un valor inicial o no. Las variables dentro de un mismo ámbito no pueden tener el mismo identificador. `Asterix` permite también la creacion de variables globales. Para declarar una variable como global, se usa la palabra reservada global y se escriben fuera de la función main. Empleamos el siguiente ejemplo para ilustrar su uso cuando las declaramos sin inicializar:

```
global intix numero_global;
intix num1;
floatix num2;
enumix dias;
charix a;
vectix<vectix<intix>[3]>[3] matriz;
```

Estas variables sin inicializar, nosotros las vamos a guardar con un valor. Los intix se inicializan con el valor inicial 0, los floatix con 0.0 y los charix con el caracter ASCII 'a'.

Caracteres especiales

`Asterix` permite al programador añadir comentarios de una línea al código. Estos comentarios son ignorados en el proceso de compilación junto con los espacios, tabulaciones y saltos de línea. Los comentarios comienzan siempre con el identificador // y se ignora todo lo que haya escrito hasta el siguiente salto de línea. Como ya avanzamos en la sección anterior, empleamos el caracter ; para separar las instrucciones del programa. Mostramos ahora unos ejemplos de comentario para ayudar a comprender su uso:

```
// Esto es un comentario
int variable; // Esta variable representa ...
```

Funciones

`Asterix` permite dividir el trabajo que hace un programa en tareas más pequeñas separadas de la parte principal. Estas tareas son lo que conocermos como pociones. Todo programa escrito en `Asterix` comienza su ejecución en la función panoramix[2] y es ella la que llama al resto de procedimientos. Las funciones siguen el siguiente esquema:

```
poc nombre_funcion(arg1 : type1,.., argn : typen) -> typer {
    // Cuerpo de la función
}
```

Todas las funciones se declaran con la palabra reservada `poc`. La flecha (`->`) junto con `typer` indican el tipo del valor que devuelve la función, en su ausencia, significa que la función no devuelve ningún valor. Las variables `argi` son los parámetros que recibe la función y vienen escritas junto con su tipo `typei`. Si estamos pasando ese arguento por referencia, escribimos `&arg`.

Instrucciones

En esta sección presentamos las instrucciones del lenguaje. Algunas de estas instrucciones dependen de una condición que ha de ser una sentencia que devuelva un valor booleanix, y esta puede ser el valor booleanix sí (galo), (true) si la condición se cumple, o romano si esta no se cumple (false). También puede contener el nombre de una variable booleanix, y el valor de la expresión dependerá de su contenido. Se debe tener en cuenta que además de las variables también puede haber llamadas a pociones que devuelvan un valor.

Instrucciones condicionales

`Asterix` admite la estructura de bifurcación condicional, para determinar que acciones tomar dada o no cierta condición. La sintaxis que sigue la estructura if-then-else es la siguiente:

```
if ( condición ) {
    // Acciones a realizar si se cumple la condición.
}
else {
    // Acciones a realizar si no se cumple la condición.
}
```

Para poder diferenciar unas con otras se obliga que se pongan llaves para indicar el ámbito de cada estructura.

Bucle while

El bucle while es un ciclo repetitivo basado en los resultados de una expresión lógica . El propósito es repetir un bloque de código mientras una condición se mantenga verdadera. La sintaxis que sigue esta estructura es:

[2]Panorámix, el druida. Creador de la poción mágica, el hombre más sabio del pueblo. En el primer libro, tiene su casa al lado de un manantial, y su cabaña tiene un palomar y una gran chimenea. Su nombre proviene de panoramique (panorámico).

```
while ( condición ) {
    // Acciones a realizar mientras se cumpla la condición.
}
```

Más instrucciones del lenguaje

`Asterix` viene con más instrucciones por defecto:

- `return X`: Es obligatoria al final de las pociones que devuelven un valor de un cierto tipo. Indica el valor que se devuelve.
- `skip`: Instrucción vacía. El programa continua con la siguiente instrucción.
- `input()`: Esta función permite la entrada de cadenas de texto por consola.
- `print( vect<charix>() )`: Esta función permite la salida de cadenas de texto por consola.

Gestión de errores

`Asterix` proporciona una gestión de errores básica indicando la línea y el motivo del error. El compilador intentará seguir con el análisis sintántico y semántico buscando posibles errores posteriores.

Ejemplo de código

El primer ejemplo consiste en un programa que multiplica dos matrices 3x3.

```
proc panoramix() {
    vect<vect<intix>[3]>[3] mat1;
    vect<vect<intix>[3]>[3] mat2;

    multmatrix(ma)
}

proc multmatrix(mat1 : vect<vect<intix>[3]>[3],
                mat2 : vect<vect<intix>[3]>[3]) -> vect<vect<intix>[3]>[3] {
    while

}
```