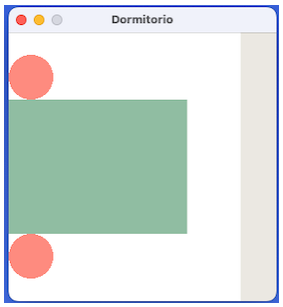
AMUEBLA: un lenguaje para amueblar habitaciones

Descripción del lenguaje

Este proyecto consiste en la construcción de un traductor para un lenguaje llamado AMUEBLA.

El traductor recibirá un programa escrito en el lenguaje AMUEBLA (con extensión .amu), que permite definir plantas de habitaciones con recuadros y círculos que representan muebles.

El resultado será un fichero con un programa escrito en C++ con llamadas a la librería AMUEBLA que, al compilarse y ejecutarse, mostrará en pantalla una secuencia de habitaciones según la definición que aparezca en el fichero de entrada.



Un programa escrito en el lenguaje AMUEBLA estará formado por una serie de bloques consecutivos donde se definen muebles, variables y una secuencia de habitaciones.

¡Atención! Sobre valores, expresiones y límites

Siempre que se haga referencia a un valor entero, real o bool en esta descripción del lenguaje, debe entenderse que se puede usar una expresión de tipo entero, real o bool para conseguir ese valor.

Solo se pueden usar variables dentro de las expresiones numéricas (enteras o reales), y no en las lógicas (igual que en la actividad 6).

Podemos suponer que los identificadores y cadenas tendrán una longitud inferior siempre a 100 caracteres. No es necesario comprobarlo.

Se puede suponer, en general, que nunca habrá más de 100 elementos de cualquier tipo (identificadores, variables, habitaciones, etc.)

Representación gráfica

El lenguaje AMUEBLA está pensado para describir muebles geométricos que se representarán en una ventana de, como máximo, 1000 x 1000 unidades que se pueden colorear individualmente (cada unidad podría representar, por ejemplo, un centímetro de un plano).

Cada una de las posiciones viene determinada por su fila y su columna (f, c). La esquina superior izquierda es la posición (0, 0), y la esquina inferior derecha, la (999, 999).

Además, en la parte superior de la ventana se puede mostrar una cadena de caracteres, con el objetivo de dar título a la habitación correspondiente.

Algunas de las instrucciones que se explican a continuación se convertirán, en el fichero de salida, en llamadas a operaciones de la librería AMUEBLA.

Además, para que el fichero de salida se pueda compilar y se inicie y finalice correctamente el entorno gráfico, habrá que escribir una serie de instrucciones de C/C++ al principio y al final. Lo mejor es ver uno de los ejemplos que acompañan al proyecto.

1. Comentarios

En los programas escritos en AMUEBLA es posible incluir comentarios.

Los comentarios comienzan con **%%** (no necesariamente al inicio de una línea) y finalizan con el salto de línea.

2. Bloques básicos

Los programas de AMUEBLA se estructuran en varias secciones o bloques. Cada una de las secciones comienza con una palabra reservada escrita en mayúsculas y que aparece sola en la línea.

Los bloques deben aparecer en el orden que se indica a continuación.

VARIABLES

En esta sección se definirán variables de tipo entero, real o lógico que posteriormente serán utilizadas y modificadas en otras secciones del programa.

Esta sección es opcional y podría no aparecer el nombre de la sección.

Cada definición se separa de la siguiente con uno o más saltos de línea.

Puede haber tres formatos distintos de definición:

```
Tipo secuencia_de_identificadores

Tipo identificador <- expresión

identificador <- expresión
```

El Tipo será el nombre de uno de los tipos predefinidos: `Entero`, `Real` o `Bool`.

La secuencia de identificadores tendrá, como mínimo, un identificador de variable. Si son varios, irán separados por comas.

El identificador de una variable empezará por una letra minúscula, y luego puede continuar con letras mayúscula y minúsculas, dígitos y guiones bajos (_).

En el segundo formato, solo se define una variable que también se inicializa con el valor de la expresión indicada tras el operador de asignación (`<-`).

En el tercer formato, se usa la asignación para cambiar el valor de una variable definida en las líneas anteriores.

Para que las asignaciones sean correctas, el valor resultante de evaluar la **`expresión`** deberá ser del mismo tipo que el usado para definir la variable.

Las expresiones serán como las definidas en las actividades 5 y 6.

En la definición de las habitaciones se podrá actualizar el valor de una variable con una asignación, siempre que no se modifique el tipo de la variable definido en esta sección.

MUEBLES

En esta sección se definirá el aspecto de los muebles que aparecerán en las habitaciones más adelante. Esta sección es **obligatoria**.

La definición de cada mueble se hará en una línea con uno de los siguientes formatos:

```
Nombre = <rectangulo, alto, ancho, color>

Nombre = <circulo, radio, color>
```

`rectangulo` y **`circulo`** son palabras reservadas del lenguaje que indican la forma del mueble.

`Nombre` es un identificador que debe empezar por una letra mayúscula, y luego puede continuar con letras mayúscula y minúsculas, dígitos y guiones bajos (_).

`alto`, `ancho` y `radio` son expresiones de tipo numérico (entero o real). (Podrían incluir variables definidas en el bloque inicial.)

`color` será uno de los siguientes colores, escritos en minúscula: `negro`, `gris`, `rojo`, `azul`, `amarillo`, `verde`, `marrOn`.

3. Definición de habitaciones

Tras los bloques anteriores, aparecerá en el programa una secuencia de definiciones de habitaciones. Como mínimo debe haber la definición de una habitación.

Cada definición empieza con una línea con el siguiente formato:

```
HABITACION ( alto , ancho ) Cadena :
```

La palabra reservada HABITACION debe ir seguida de dos números enteros (`alto` y `ancho` de la ventana) entre paréntesis separados por una coma y una cadena entre comillas y dos puntos (:) y un salto de línea.

En el fichero de salida, esta definición se convertirá en una llamada a la operación `nuevaHabitacionAmu` con `alto`, `ancho` y `Cadena` como parámetros.

La definición de una habitación termina con la siguiente palabra reservada en una línea:

```
FINHABITACION
```

Esta palabra reservada se convertirá siempre en el fichero de salida en una llamada a la operación de la librería `pausaAmu` con parámetro 1.5.

En el siguiente apartado se presentan las instrucciones que pueden formar parte de la definición de una habitación.

4. Instrucciones en la definición de las habitaciones

A continuación, se detallan las distintas instrucciones que se pueden utilizar para definir una habitación.

Cada instrucción simple se separa de la siguiente por un salto de línea.

situar

La instrucción `situar` sirve para dibujar un mueble en una posición de la habitación. El formato es el siguiente:

```
situar ( Nombre, fila, col)
```

`Nombre` será el identificador de un mueble definido en el bloque `MUEBLES`.

`fila` y `col` son expresiones numéricas enteras que representan una fila o columna de la ventana que representa la habitación.

Si lo que se va a pintar es un mueble con forma de recuadro, la posición (fila, col) hace referencia a la esquina superior izquierda.

Si lo que se va a pintar en un mueble con forma de círculo, la posición (fila, col) hace referencia al centro de la circunferencia.

Cada instrucción `situar` se traducirá por una instrucción `rectanguloAmu` o `circuloAmu` en el fichero de salida.

pausa

La instrucción `pausa` sirve para detener la ejecución durante una cantidad de segundos. Tiene el siguiente formato:

```
pausa ( tiempo )
```

`tiempo` es una expresión numérica **real** o **entera** que representa el número de segundos.

Cada instrucción `pausa` se traducirá por una instrucción `pausaAmu` en el fichero de salida.

mensaje

La instrucción `mensaje` sirve para mostrar un comentario en el fichero de salida. Tiene el siguiente formato:

```
mensaje ( cadena )
```

`cadena` es una cadena de texto entre comillas dobles que representa el texto que se escribirá en la consola.

Cada instrucción `mensaje` se traducirá por un comentario de una línea (con //) en el fichero de salida escrito en lenguaje C++.

asignación

En la definición de una habitación se podrá cambiar el valor de las variables definidas en la sección de variables.

El formato será el siguiente:

```
identificador <- expresión
```

Si se cambia el valor dentro de la definición de una habitación, el valor se cambia para cualquier uso posterior, en la misma o en distinta habitación. (No se aplica ninguna regla de ámbito de validez de las asignaciones.)

El uso de esta instrucción tiene las mismas características y limitaciones que las asignaciones explicadas en la sección de variables.

5. Estructuras de control

Además de las instrucciones simples del apartado anterior, en la definición de una habitación se podrán incluir estructuras de control (alternativas y bucles).

Estas estructuras de control incluirán bloques de instrucciones.

Estructura alternativa o condicional

Esta estructura de control puede tener dos formatos distintos:

```
si (condición) {
            secuencia_de_instrucciones
}
```

`condición` es una expresión booleana y `secuencia_de_instrucciones`, una secuencia de instrucciones (simples o estructuras de control). En este caso, las instrucciones se ejecutarán solo si la condición es cierta.

El otro formato es el siguiente:

```
si (condición) {
            secuencia_de_instrucciones1
}
si_no {
```

```
            secuencia_de_instrucciones2
    }
```

En este caso, `secuencia_de_instrucciones1` se ejecutarán si la condición es cierta y `secuencia_de_instrucciones2` si la condición es falsa.

`si, si_no` son palabras reservadas del lenguaje, y la secuencia de instrucciones va entre llaves.

Bucle

En AMUEBLA hay un único tipo de bucle cuya sintaxis es:

```
repetir n {
     secuencia_de_instrucciones
}
```

`n` es una expresión numérica de tipo entero e indica el número de veces que se repiten las instrucciones que hay dentro del bucle.

6. Espacios en blanco y saltos de línea

Aunque en estos ejemplos se han separado muchos de los elementos con espacios en blanco (por ejemplo, los paréntesis llevan siempre un espacio delante y otro detrás), eso no es necesario o puede haber varios más de los indicados. Los espacios solo importan en aquellos casos en los que se pueda malinterpretar una cadena en el analizador léxico. (Por ejemplo, no se puede separar < y = para indicar el operador <=; el tipo Entero debe separarse del identificador posterior para que no se considere un único identificador.)

Además de los saltos de línea obligatorios (por ejemplo, tras la definición de una instrucción), puede haber líneas en blanco adicionales en el programa de entrada (al principio, tras cada instrucción, etc.)

7. Errores sintácticos y semánticos

Siempre que se detecte un error sintáctico o semántico se mostrará un mensaje de error en la consola. Aunque el programa fuente contenga errores, se intentará generar el fichero de salida.

Se deben tener en cuenta todos los errores semánticos indicados en las actividades de la calculadora y los que se desprendan del enunciado del proyecto.