PROYECTO AMUEBLA

TEORÍA DE LENGUAJES 23-24

Amuebla | Documentación del Proyecto

➔ **Autores:** Dario Alvarez Barrado, Ignacio Alcalde Torrescusa

➔ **Fecha:** Junio 2024

ÍNDICE

1. Introducción

El proyecto Amuebla se centra en la implementación de un sistema de gestión de muebles que permite gestionar y visualizar muebles en un entorno gráfico. Utiliza diversas estructuras de datos y funciones auxiliares para manejar la información relacionada con los muebles y su representación visual. El objetivo principal es proporcionar una herramienta eficiente y fácil de usar para la gestión de muebles, permitiendo operaciones como la creación, modificación y eliminación de elementos. Además, se emplean técnicas de análisis léxico y sintáctico para procesar comandos y datos de entrada, haciendo el proyecto más robusto y flexible.

2. Estructura del Proyecto

La estructura del proyecto está organizada en varios archivos y carpetas, cada uno con una función específica. Los componentes principales incluyen:

- **Archivos de análisis léxico y sintáctico:** ‘expresiones.c’, ‘expresiones.y’, ‘lex.yy.c’, ‘lexico.l’. Estos archivos contienen las definiciones y reglas para el análisis de expresiones y la generación de tokens, utilizando herramientas como Bison y Flex.

- **Archivos de estructura de datos:** ‘estructura.h’, ‘estructura.cpp’, ‘estructuraMuebles.h’, ‘estructuraMuebles.cpp’. Estos archivos definen y manejan las estructuras de datos necesarias para gestionar los identificadores y los muebles.

- **Archivos de entrada y salida:** ‘basica.amu’, ‘basica.cpp’. Contienen los datos de entrada para el programa y los resultados generados.

- **Archivos de configuración de compilación:** ‘makefile’, ‘makeAmuebla’. Estos archivos contienen las instrucciones para compilar y ejecutar el proyecto de manera automatizada.

- **Archivos de implementación principal:** ‘amuebla.cpp’, ‘amuebla.h’. Gestionan la lógica del programa y la interfaz gráfica utilizando la biblioteca Allegro.

La estructura modular del proyecto facilita la gestión del código y la colaboración, permitiendo trabajar en diferentes componentes de manera independiente y eficiente.

3. Diseño de Estructuras de Datos

Tabla de Identificadores
La clase IdentifiersTable gestiona una colección de identificadores, cada uno representado por la estructura IdentifierInfo. Esta estructura incluye el identificador (una cadena de caracteres), un tipo (un entero que puede representar diferentes tipos de datos), y un valor almacenado en una unión tipo_valor que puede ser un entero, un número real, un booleano o una cadena de caracteres.

La clase proporciona métodos para añadir un identificador a la colección (addIdentifier), imprimir la información de todos los identificadores en un archivo (printIdentifiersInfo), y buscar un identificador por su nombre (searchIdentifier). La colección de identificadores se almacena en un vector privado de IdentifierInfo. Se decidió usar una estructura de vector para almacenar los identificadores debido a su eficiencia y facilidad de uso en C++. Esto permite una búsqueda rápida y una gestión dinámica de los identificadores.

Estructuras de Muebles
La clase MueblesTable se encarga de gestionar una colección de muebles, cada uno representado por la estructura MuebleInfo. Esta estructura contiene información detallada del mueble, incluyendo su nombre, tipo (rectángulo, círculo, o no encontrado), valores específicos (como el ancho y altura para rectángulos, o el radio para círculos), y su color. Los valores pueden ser enteros o reales, manejados mediante una unión tipo_valor_mueble y un enumerador TipoValor para especificar el tipo de dato.

La clase proporciona métodos para agregar un mueble a la colección (addMueble), imprimir la información de todos los muebles en un archivo (printMueblesInfo), buscar un mueble por su nombre (buscarMueblePorNombre), y obtener una referencia constante a la lista de muebles (getMuebles). La colección de muebles se almacena en un vector privado de MuebleInfo, asegurando una gestión eficiente y organizada de los datos.

4. Compilación y Ejecución

Para ejecutar el proyecto, primero utiliza el comando make para generar el fichero basica.cpp y su correspondiente ejecutable. Luego, emplea el comando make -f makeAmuebla para compilar el proyecto completo utilizando la biblioteca Allegro, lo cual incluye la ejecución de la interfaz gráfica.

5. Errores Detectados

En cuanto a los errores detectados tuvimos problemas para establecer la estructura que seguían los bloques de variables, muebles y habitaciones. Sin embargo, una vez solucionado esto, crear la estructura de muebles no fue difícil ya que ya habíamos creado la de identificadores. Además, enfrentamos problemas en la identificación de condicionales y en la ejecución o no de las instrucciones según se cumplían o no las expresiones. Para solucionar esto, duplicamos el bloque de instrucciones que puede haber en una habitación e incorporamos una variable booleana para identificar cuándo se tenían que ejecutar las instrucciones o no.

6. Conclusiones

El proyecto Amuebla proporciona una herramienta eficiente para la gestión y visualización de muebles en un entorno gráfico. A lo largo del desarrollo del proyecto, se adquirieron conocimientos valiosos sobre la gestión de estructuras de datos, el análisis léxico y sintáctico, y la integración de bibliotecas gráficas. Trabajamos ambos por igual en todas las etapas del proyecto y nos ha gustado mucho esta experiencia.