Manual Técnico

Irving Alvarado

Introducción

Este programa está diseñado para gestionar y ordenar listas de números ingresados por el usuario o l eídos desde un archivo CSV. Utiliza varios algoritmos de ordenamiento, como el Ordenamiento por Burbuja, el Ordenamiento Rápido (Quick Sort) y el Ordenamiento por Inserción (Insertion Sort). A demás, el programa puede generar un reporte detallado del proceso de ordenamiento, incluyendo el estado del arreglo antes y después de cada algoritmo.

Objetivos y Alcance

El objetivo principal de este programa es ofrecer una herramienta educativa y práctica para entender y visualizar distintos algoritmos de ordenamiento. Permite a los usuarios ingresar datos manualment e o a través de un archivo CSV y elegir el tipo de ordenamiento a aplicar. El alcance del programa i ncluye:

- Ingreso de datos por teclado.
- Ingreso de datos desde archivos CSV.
- Aplicación de distintos algoritmos de ordenamiento.
- Generación de reportes detallados en archivos de texto.
- Manejo de datos de manera dinámica y eficiente.

Requisitos del Sistema

Para ejecutar este programa, se requiere un entorno de desarrollo compatible con ARM64, como QE MU para emulación, y un ensamblador ARM64, como `aarch64-linux-gnu-as`, para compil ar el código ensamblador. Además, es necesario tener configurado un entorno de terminal para inter actuar con el programa.

Sección de datos

Usted

Ahora la sección de datos

Copilot

Sección de Datos

Definiciones de Variables y Buffers

En la sección de datos de este programa, se definen todas las variables y buffers necesarios para el manejo de la entrada de datos, almacenamiento temporal y manipulación de cadenas. Aquí se declar an las variables utilizadas para la gestión de los datos de entrada y salida, así como los buffers para el procesamiento interno.

- **Variables Principales**:
    - `opcion`: Espacio reservado para almacenar la opción seleccionada por el usuario en el menú principal.
    - `filename`: Buffer para almacenar el nombre del archivo CSV que se leerá.
    - `count`: Variable que almacena la cantidad de números leídos desde el archivo CSV o ingresados manualmente.
    - `cadena`: Buffer utilizado para almacenar la cadena de números ingresados manualm ente.
    - `num`: Buffer temporal para almacenar un número durante la conversión de ASCII a e ntero.
    - `character`: Variable para almacenar un carácter leído desde la entrada estándar.
    - `fileDescriptor`: Espacio reservado para el descriptor de archivo durante la oper ación de lectura/escritura de archivos.
    - `array`: Buffer principal para almacenar los números ingresados o leídos.
    - `length`: Variable utilizada para almacenar la longitud de una cadena.
    - `stepCounter`: Contador de pasos utilizado durante los algoritmos de ordenamient o.
- **Buffers de Cadenas y Mensajes**:
    - `reporteNombre`: Nombre del archivo de reporte ("reporte.txt").
    - `titulo`: Buffer para almacenar el título del ordenamiento actual.
    - `buffer`: Buffer general para almacenar datos temporales convertidos a ASCII.
    - `encabezado`: String que contiene el encabezado del reporte, incluyendo informaci ón del autor, curso y detalles.

- `ordenBurbuja`, `ordenQuickSort`, `ordenInsertion`: Strings que contienen los nombres de los diferentes algoritmos de ordenamiento.

Inicialización de Datos

Al inicio del programa, las variables y buffers se inicializan para asegurarse de que están en un esta do adecuado antes de cualquier operación. Esto se logra mediante la sección `.bss`, donde las varia bles se inicializan con ceros o espacios reservados para datos temporales.

- **Estructura** `.bss`:
    - Define las variables que no requieren un valor inicial específico.
    - Utiliza `.space` y `.zero` para reservar espacio en memoria para las variables.

Esta sección proporciona una base sólida para el manejo y manipulación de datos a lo largo del prog rama. La correcta definición y inicialización de estas variables son cruciales para el funcionamiento eficiente y preciso del programa.

Funciones principales

`_start`: Punto de Entrada del Programa

La función `_start` es el punto de entrada principal del programa. Inicializa el entorno, limpia la pantalla, imprime el encabezado, y luego dirige el flujo del programa al menú principal. Esta funció n asegura que todo esté listo antes de que el usuario interactúe con el programa.

`menu`: Menú Principal

La función `menu` despliega las opciones principales para el usuario, permitiéndole seleccionar entr e ingresar una lista de números, ordenar los números utilizando distintos algoritmos, o finalizar el pr ograma. Esta función es crucial para la navegación del programa y la interacción del usuario.

`seleccion_ingreso`: Selección del Método de Ingreso

La función `seleccion_ingreso` permite al usuario elegir entre ingresar los números manualm ente a través del teclado o cargar los números desde un archivo CSV. Esta selección es fundamental para adaptar la entrada de datos al método preferido por el usuario.

`tipoOrdenamientoBubble`: Selección del Ordenamiento de Burbuja

La función `tipoOrdenamientoBubble` ofrece al usuario la opción de ordenar los números en orden ascendente o descendente usando el algoritmo de Burbuja. Esta función establece una bander a que determina el tipo de ordenamiento a aplicar.

`metodoTeclado`: Ingreso de Datos por Teclado

La función `metodoTeclado` maneja la entrada manual de números por parte del usuario. Solicit a al usuario que ingrese los números separados por comas, los procesa y almacena en el buffer princ ipal (`array`). Esta función permite un método de ingreso de datos directo y flexible.

`metodoCSV`: Ingreso de Datos desde un Archivo CSV

La función `metodoCSV` gestiona la lectura de números desde un archivo CSV. Solicita el nombre del archivo, lo abre, lee los números y los almacena en el buffer principal (`array`). Esta función f acilita la carga de datos desde fuentes externas, asegurando una mayor flexibilidad para el usuario.

`bubbleSort`: Ordenamiento por Burbuja

La función `bubbleSort` implementa el algoritmo de ordenamiento por burbuja. Puede ordenar lo s números en orden ascendente o descendente, dependiendo de la selección del usuario. Durante el proceso, imprime el estado del arreglo en cada paso, permitiendo al usuario visualizar el progreso d el ordenamiento.

`quickSort`: Ordenamiento Rápido

La función `quickSort` implementa el algoritmo de ordenamiento rápido. Utiliza una función aux iliar `partition` para dividir el arreglo en partes y ordenarlas recursivamente. Este algoritmo es e ficiente para grandes conjuntos de datos y también puede ordenar en ambos sentidos (ascendente y descendente).

`insertionSort`: Ordenamiento por Inserción

La función `insertionSort` aplica el algoritmo de ordenamiento por inserción. Ordena los núm eros de forma ascendente o descendente, según la selección del usuario. Esta función también impri me el estado del arreglo en cada paso, facilitando la comprensión del proceso de ordenamiento.

`clear_input_buffer`: Limpieza del Buffer de Entrada

La función `clear_input_buffer` asegura que cualquier entrada residual en el buffer se elimin e antes de procesar nuevas entradas. Esto previene errores y garantiza la precisión de los datos ingre sados.

`clear_array`: Limpieza del Array de Números

La función `clear_array` inicializa o limpia el buffer principal (`array`), asegurando que esté li bre de datos residuales antes de cargar nuevos números.

`atoi`: Conversión de ASCII a Entero

La función `atoi` convierte cadenas ASCII a números enteros. Es esencial para procesar las entrad as de texto del usuario y convertirlas en datos numéricos que el programa pueda manipular.

`itoa`: Conversión de Entero a ASCII

La función `itoa` convierte números enteros a cadenas ASCII. Esto es vital para la impresión de n úmeros y la generación de reportes, permitiendo que los datos numéricos sean fácilmente legibles y comprensibles.

`copiarCadena`: Copia de Cadenas

La función `copiarCadena` copia el contenido de una cadena origen a una cadena destino. Es útil para gestionar títulos y mensajes dentro del programa, asegurando que las cadenas se manipulen cor rectamente.

`array_to_ascii`: Conversión del Array a ASCII para Impresión

La función `array_to_ascii` convierte el contenido del buffer principal (`array`) a formato A SCII para impresión. Esto permite que los números se presenten de manera legible en la consola y e n los reportes generados.

Funciones de Archivo

`openFile`: Apertura de Archivos

La función `openFile` maneja la apertura de archivos, estableciendo los permisos necesarios y ase gurando que el archivo esté listo para operaciones de lectura o escritura.

`closeFile`: Cierre de Archivos

La función `closeFile` cierra un archivo abierto, liberando cualquier recurso asociado con el des criptor de archivo. Esto es esencial para evitar fugas de memoria y asegurar que los datos se guarde n correctamente.

`readCSV`: Lectura de Archivos CSV

La función `readCSV` lee números desde un archivo CSV y los almacena en el buffer principal (`a rray`). Procesa cada línea del archivo, asegurando que los datos se carguen correctamente.

`escribirReporte`: Escritura de Reportes en Archivo

La función `escribirReporte` genera un reporte en un archivo de texto, documentando el título del ordenamiento y el estado del arreglo antes y después del proceso de ordenamiento. Esto permite un registro detallado del proceso y los resultados.

Funciones auxiliares

`clear_input_buffer`: Limpieza del Buffer de Entrada

La función `clear_input_buffer` es crucial para asegurar que cualquier entrada residual en el buffer de entrada sea eliminada antes de procesar nuevas entradas. Esto previene la mezcla de datos y asegura la precisión de las entradas del usuario. Esta función se llama después de leer cada opción del usuario para garantizar que no queden caracteres residuales que puedan interferir con el procesa miento posterior.

`clear_array`: Limpieza del Array de Números

`clear_array` inicializa o limpia el buffer principal (`array`) que se utiliza para almacenar los números ingresados o leídos. Esta función es esencial antes de cargar nuevos datos para asegurar qu e el array esté libre de datos previos que puedan causar inconsistencias. La función recorre el array y establece todos los valores a cero.

`atoi`: Conversión de ASCII a Entero

La función `atoi` convierte cadenas ASCII que representan números en sus valores enteros corresp ondientes. Es una función auxiliar vital para procesar las entradas de texto del usuario y convertirlas en datos numéricos utilizables dentro del programa. Esto permite que el programa maneje tanto la e ntrada de datos manual como la lectura de archivos CSV.

`itoa`: Conversión de Entero a ASCII

`itoa` convierte valores enteros en sus representaciones en formato de cadena ASCII. Esta convers ión es crucial para la impresión de números y la generación de reportes. Permite que los datos numé ricos se presenten de manera legible y comprensible para el usuario, y es utilizada en varias partes d el programa para mostrar los resultados de los algoritmos de ordenamiento.

`copiarCadena`: Copia de Cadenas

La función `copiarCadena` copia el contenido de una cadena de origen a una cadena de destino. Se utiliza para gestionar títulos y mensajes dentro del programa, asegurando que las cadenas se man ipulen correctamente y se puedan mostrar o almacenar sin problemas. Es especialmente útil para co nfigurar títulos y encabezados en los reportes generados.

`array_to_ascii`: Conversión del Array a ASCII para Impresión

`array_to_ascii` convierte el contenido del buffer principal (`array`) a formato ASCII para s u impresión. Esta función permite que los números del array se presenten de manera legible en la co nsola y en los reportes generados. La conversión asegura que los datos numéricos puedan ser fácilm ente entendidos y revisados por el usuario.

`copiarCadenaBuffer`: Copia de Datos entre Buffers

La función `copiarCadenaBuffer` es una variante de `copiarCadena` que copia el contenid o de un buffer de origen a un buffer de destino. Esta función se utiliza para manejar datos temporale

s dentro del programa, asegurando que la información se almacene y se transfiera de manera eficien te y sin pérdidas.

Estas funciones auxiliares son esenciales para el funcionamiento fluido y eficiente del programa. Pr oveen utilidades críticas para la manipulación de datos, conversión de formatos, y manejo de entrad as y salidas. Gracias a estas funciones, el programa puede manejar una variedad de tareas y proceso s de manera robusta y confiable.

Conclusiones

Resumen del Funcionamiento del Programa

Este programa es una herramienta completa y versátil diseñada para manejar y ordenar listas de nú meros ingresados por el usuario o leídos desde un archivo CSV. Mediante el uso de algoritmos de or denamiento como Burbuja, Quick Sort, y Inserción, el programa no solo realiza las operaciones de manera eficiente, sino que también proporciona una visualización detallada del proceso, mostrando el estado del arreglo en cada paso. Esta funcionalidad es especialmente útil en contextos educativos y de aprendizaje, donde es crucial entender cómo funcionan los algoritmos internamente.

Funcionalidades Clave

1. **Ingreso de Datos Flexible**: Los usuarios pueden ingresar datos manualmente a través del te clado o cargar un archivo CSV. Esta flexibilidad permite al programa adaptarse a diferentes necesidades y preferencias.
2. **Variedad de Algoritmos de Ordenamiento**: Con la implementación de Burbuja, Quick Sor t y Inserción, el programa ofrece una amplia gama de opciones para ordenar los datos, cada una adecuada para diferentes tipos de conjuntos de datos y escenarios de uso.
3. **Generación de Reportes Detallados**: El programa puede generar reportes en archivos de te xto, documentando el proceso de ordenamiento y proporcionando un registro detallado del e stado del arreglo antes y después de la ordenación. Esto facilita el seguimiento y análisis de l os resultados.

El desarrollo de este programa ha proporcionado una experiencia rica en aprendizaje, destacando la importancia de la organización del código, la correcta gestión de la memoria y la claridad en la impl ementación de algoritmos complejos. Cada función y subrutina ha sido diseñada cuidadosamente pa ra asegurar un rendimiento óptimo y una interacción eficiente con el usuario.

Para futuras mejoras, se podrían considerar las siguientes recomendaciones:

1. **Optimización de Algoritmos**: Aunque los algoritmos implementados son efectivos, siempr e hay espacio para optimizaciones adicionales que podrían mejorar el rendimiento en conjun tos de datos muy grandes.
2. **Interfaz de Usuario Mejorada**: Desarrollar una interfaz de usuario más intuitiva y amigabl e podría hacer que el programa sea aún más accesible y fácil de usar para una audiencia más amplia.
3. **Ampliación de Funcionalidades**: Incorporar más algoritmos de ordenamiento y permitir co mparaciones directas entre ellos podría enriquecer aún más el valor educativo del programa.
4. **Validación y Manejo de Errores**: Mejorar la validación de la entrada de datos y el manejo de errores aseguraría que el programa maneje una mayor variedad de casos de uso y condici ones inesperadas con gracia y robustez.