Documentación de Lista Enlazada

Jeronimo Rojano Kevin Marzul

Antonio Ramírez Juan Elías

Nava Soler Raúl

Introducción

El problema a resolver es la implementación de una lista utilizando arreglos en lenguaje C. Este tipo de estructura es útil en diversos contextos, como la gestión de memoria, el control de recursión y la implementación de algoritmos. La decisión de no utilizar memoria dinámica ni librerías adicionales se tomó para mantener el código lo más simple y accesible posible, permitiendo su ejecución en prácticamente cualquier entorno que soporte C. Esta limitación, sin embargo, introduce ciertas restricciones, como el tamaño fijo de la lista.

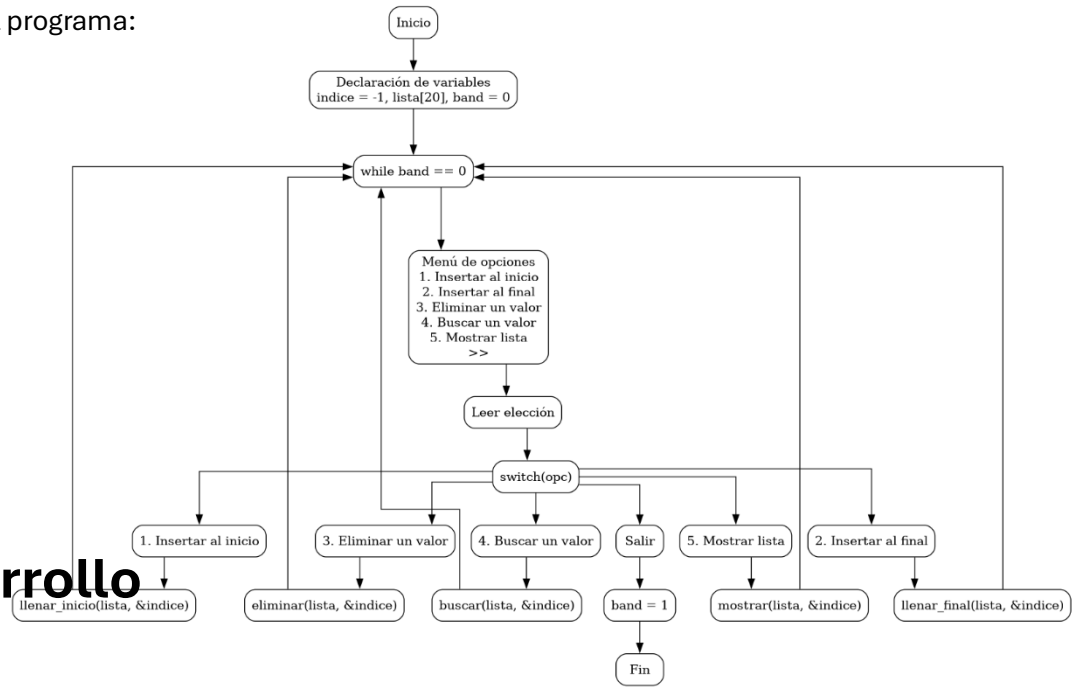
Métodos y Materiales

La implementación se realiza en un archivo de código fuente en C. La estructura de datos utilizada es un arreglo estático de tamaño fijo. Se definen varias funciones para manipular la lista, incluyendo la inserción de elementos al inicio y al final, la eliminación de elementos, la búsqueda de elementos y la visualización de la lista.

- **Herramientas Utilizadas Lenguaje de Programación: C / Version C23**
- **Compilador:** gcc.exe (tdm64-1) 10.3.0 Copyright (C) 2020 Free Software Foundation, Inc. Esto es software libre; vea el codigo para las condiciones de copia. NO hay garantia; ni siquiera para MERCANTIBILIDAD o IDONEIDAD PARA UN PROPOSITO EN PARTICULAR
- **Entorno de Desarrollo:** Visual Studio Code

Descripción General del Código

El código está compuesto por una estructura de datos y varias funciones que operan sobre esta estructura. A continuación, se presenta un diagrama de flujo que ilustra el funcionamiento general del programa:



Desarrollo

Funciones Implementadas

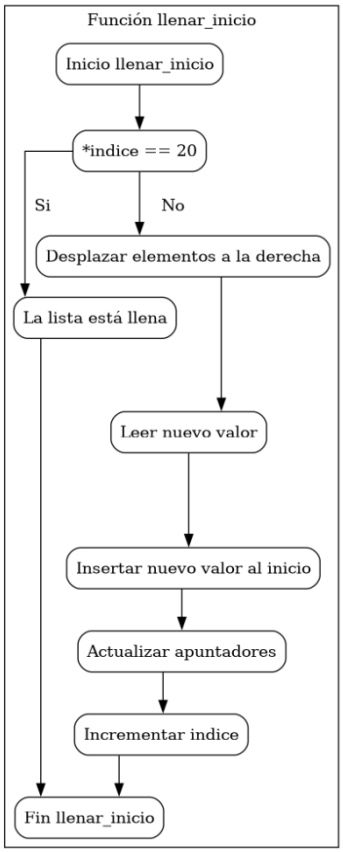
Se implementaron las siguientes funciones para manipular la lista:

1. **llenar_inicio**: Inserta un nuevo valor al inicio de la lista.
2. **llenar_final**: Inserta un nuevo valor al final de la lista.
3. **eliminar**: Elimina un valor de la lista en una posición específica.
4. **buscar**: Busca un valor en la lista.
5. **mostrar**: Muestra todos los elementos de la lista.

Cada función se explica en detalle a continuación:

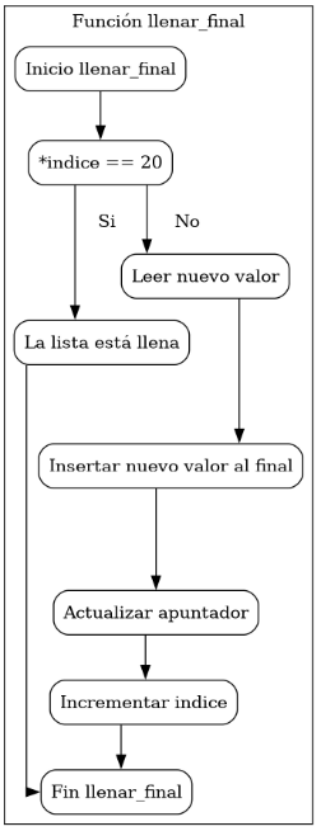
Función llenar_inicio

Esta función agrega un nuevo valor al inicio de la lista. Si la lista está llena, muestra un mensaje de error. En caso contrario, desplaza todos los elementos hacia la derecha y agrega el nuevo valor al inicio.



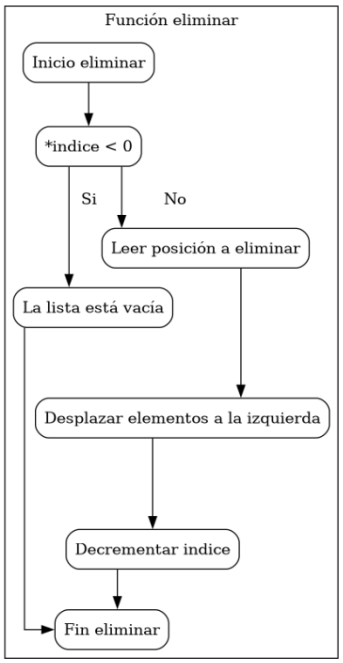
Función llenar_final

Esta función agrega un nuevo valor al final de la lista. Si la lista está llena, muestra un mensaje de error. En caso contrario, agrega el nuevo valor al final y actualiza el índice.



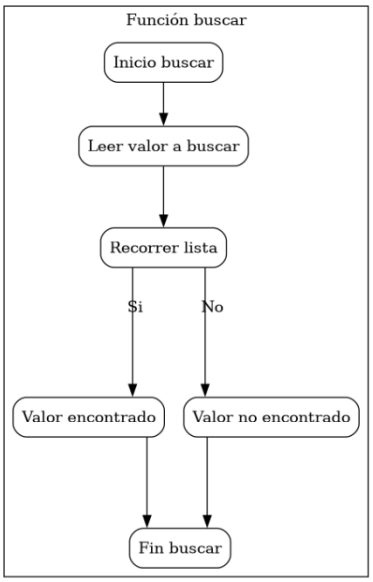
Función eliminar

Esta función elimina un valor de la lista en una posición específica. Si la lista está vacía, muestra un mensaje de error. En caso contrario, desplaza todos los elementos desde la posición eliminada hacia la izquierda y actualiza el índice.



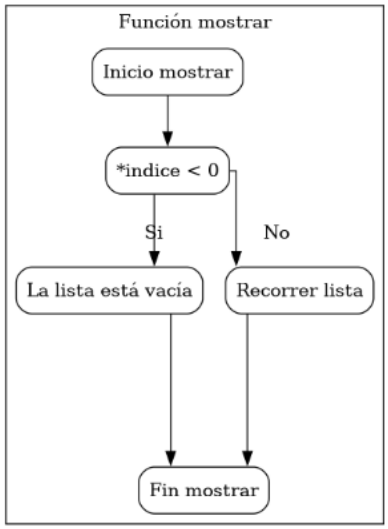
Función buscar

Esta función busca un valor en la lista. Recorre todos los elementos de la lista y verifica si el valor buscado está presente. Si se encuentra, muestra la posición del valor.



Función mostrar

Esta función muestra todos los elementos de la lista. Recorre todos los elementos de la lista y muestra su valor y la dirección a la que apuntan.



Conclusión

El programa cumple con la funcionalidad básica de una lista utilizando arreglos en C. Permite insertar elementos al inicio y al final, eliminar elementos, buscar elementos y mostrar la lista. A continuación, se presenta una tabla con los resultados de las operaciones realizadas

Operación	Resultado
Insertar al inicio	Éxito
Insertar al final	Éxito
Eliminar un valor	Éxito
Buscar un valor	Éxito (si el valor existe)
Mostrar lista	Lista completa mostrada

La implementación utiliza un arreglo estático de tamaño fijo, lo cual limita la capacidad de la lista a 20 elementos. Esta limitación se debe a la decisión de no utilizar memoria dinámica. Además, la función eliminar desplaza todos los elementos desde la posición eliminada hacia la izquierda, lo cual puede ser ineficiente para grandes cantidades de datos.

Limitaciones

- **Capacidad Fija:** La lista tiene una capacidad fija de 20 elementos, lo cual puede ser insuficiente para algunas aplicaciones.
- **Eficiencia:** La operación de eliminación puede ser ineficiente para listas grandes, ya que desplaza todos los elementos hacia la izquierda.
- **Flexibilidad:** Al no utilizar memoria dinámica, la implementación carece de flexibilidad para adaptarse a diferentes tamaños de lista.

Áreas de Mejora

- **Memoria Dinámica:** Implementar la lista utilizando memoria dinámica permitiría una capacidad variable y mejoraría la flexibilidad del programa.
- **Optimización:** Optimizar la operación de eliminación para mejorar la eficiencia, quizás utilizando una estructura de datos diferente.
- **Errores y Validaciones:** Mejorar la gestión de errores y validaciones para proporcionar una experiencia de usuario más robusta.

El programa funciona correctamente dentro de sus limitaciones. La lista permite realizar todas las operaciones básicas requeridas. Sin embargo, la capacidad limitada y la ineficiencia en la eliminación de elementos son áreas que podrían mejorarse en futuras versiones. Una posible mejora sería implementar la lista utilizando memoria dinámica para permitir un tamaño variable. Además, optimizar la operación de eliminación y mejorar la gestión de errores y validaciones podrían hacer que el programa sea más eficiente y robusto.

Evidencias de su funcionamiento:

INGRESAR 10, 20 Y 30

```
posicion   0 -->   30 apuntando a la direccion    1
posicion   1 -->   20 apuntando a la direccion    2
posicion   2 -->   10 apuntando a la direccion    3
1. Insertar al inicio
2. Insertar al final
3. Eliminar un valor
4. Buscar un valor
5. mostrar lista
>>
```

INSERTAR 5 AL INICIO

```
Ingresa el nuevo valor: 5

Valor agregado con exito

1. Insertar al inicio
2. Insertar al final
3. Eliminar un valor
4. Buscar un valor
5. mostrar lista
>>5

posicion   0 -->    5 apuntando a la direccion    1
posicion   1 -->   30 apuntando a la direccion    2
posicion   2 -->   20 apuntando a la direccion    3
posicion   3 -->   10 apuntando a la direccion    4
1. Insertar al inicio
2. Insertar al final
3. Eliminar un valor
4. Buscar un valor
5. mostrar lista
```

ELIMINAR 20

```
>>5

posicion    0 -->     5 apuntando a la direccion     1
posicion    1 -->    10 apuntando a la direccion     2
posicion    2 -->    30 apuntando a la direccion     3
```

BUSCAR 30

```
>>4

Ingresa el valor que deseas buscar: 30

El valor 30 fue encontrado en la posicion --> 2
1. Insertar al inicio
2. Insertar al final
3. Eliminar un valor
4. Buscar un valor
5. mostrar lista
```

MOSTRAR LISTA

```
posicion    0 -->     5 apuntando a la direccion     1
posicion    1 -->    10 apuntando a la direccion     2
posicion    2 -->    30 apuntando a la direccion     3
1. Insertar al inicio
2. Insertar al final
3. Eliminar un valor
4. Buscar un valor
5. mostrar lista
```