Juan Nicolas Barreto Sanchez

Estructuras de Informacion

Profesor Fernel

2024

Taller 1 estructuras

1. Cuales son las principales características de la memoria estática?
2. Cuando puede ser beneficioso utilizar memoria estática en un programa?
3. Cual puede ser una desventaja de utilizar estructuras estáticas, en términos, de gestión de memoria
4. En que situaciones la asignación de memoria, dinámica es mas adecuada que la estática
5. Cuales son los riegos asociados con la asignación dinámica de memoria
6. Como se gestiona la liberación de memoria en programacion
7. Como se podría explicar el concepto de memoria estática y dinámica a alguien que esta aprendiendo sobre programacion por primera vez
8. Como afecta el uso de memoria estática al tamaño total de la memoria del programa
9. Que sucede si se intenta cambiar el tamaño de una estructura estática, despues de haber sido declarada
10. En que situación podría ocurrir una fuga de memoria en el contexto de la memoria dinámica

Solución

1) La memoria estática, como la SRAM, se caracteriza por ser rápida y no requerir actualizaciones periódicas. Aunque es más veloz que la memoria dinámica (DRAM), tiene menor densidad de almacenamiento, mayor consumo de energía y es más costosa de producir. Se utiliza en cachés y registros internos de procesadores para acceso rápido a datos.
2) La memoria estática es beneficiosa en programas cuando se necesita acceso rápido a datos, especialmente en algoritmos con cálculos intensivos, para almacenar cantidades pequeñas pero críticas de datos, cachear resultados intermedios, implementar estructuras de datos especializadas y en operaciones en tiempo real.
3)

   Desperdicio de memoria: Si la estructura estática tiene un tamaño demasiado grande para los datos reales que se utilizan, puede haber desperdicio de memoria, ya que se asigna la cantidad máxima posible incluso si no se utiliza por completo.

   Limitaciones en la escalabilidad: Al tener un tamaño fijo, las estructuras estáticas pueden no adaptarse bien a situaciones donde la cantidad de datos varía dinámicamente. Esto puede ser un problema en programas que necesitan ser escalables y manejar conjuntos de datos cambiantes.

   Mayor consumo de memoria: En comparación con estructuras de datos dinámicas que pueden crecer o decrecer según sea necesario, las estructuras estáticas pueden llevar a un mayor consumo de memoria si se asigna más de la necesaria.

4) La asignación de memoria dinámica es más adecuada que la estática cuando se necesita adaptabilidad en el tamaño de la memoria durante la ejecución del programa. Es útil en situaciones donde la cantidad de datos puede variar dinámicamente y cuando se desea evitar desperdicio de memoria asignando solo lo necesario.
5) La asignación dinámica de memoria puede llevar a fugas y fragmentación de memoria, así como a accesos no válidos si no se gestiona correctamente. También puede introducir overhead y complejidad en el código

6) La liberación de memoria en programación se gestiona utilizando funciones como `free()` en C o `delete` en C++. Es esencial liberar la memoria asignada dinámicamente cuando ya no se necesita para evitar fugas de memoria. Es importante asignar el puntero a NULL después de liberar la memoria para evitar accesos no válidos. El uso de estructuras de datos automáticas, como smart pointers en C++, puede ayudar a simplificar la gestión de memoria.

7) **Memoria Estática:** Es como una caja con compartimientos fijos de tamaño conocido antes de iniciar el programa. Cada compartimiento guarda algo específico, pero no puedes cambiar el tamaño o añadir más compartimientos fácilmente mientras ejecutas el programa.

   Memoria Dinámica: Es como tener una bolsa que puedes ajustar y modificar mientras necesitas más o menos espacio. Puedes crear compartimientos en cualquier momento y liberarlos cuando ya no los necesitas. Es más flexible, pero necesitas ser responsable de gestionarla correctamente para evitar problemas.

8) El uso de memoria estática afecta al tamaño total de la memoria del programa de manera más predecible y estática en comparación con la memoria dinámica. La memoria estática se asigna durante la compilación o la ejecución temprana del programa y tiene un tamaño fijo que se determina antes de la ejecución.

   El tamaño total de la memoria del programa aumentará directamente en proporción al tamaño de las variables estáticas, estructuras o arreglos declarados en el código. Cada vez que agregas una variable estática, estás r eservando un espacio específico en memoria para ella. Este tamaño no cambia durante la ejecución del programa, independientemente de cuántos datos reales se almacenen en esas variables.

9) No puedes cambiar el tamaño de una estructura estática después de haber sido declarada. La memoria para estructuras estáticas se asigna durante la compilación o la ejecución temprana y tiene un tamaño fijo y constante durante la ejecución del programa. Si necesitas flexibilidad en el tamaño, deberías considerar el uso de estructuras de datos dinámicas, como arreglos dinámicos o listas enlazadas.

10) Una fuga de memoria en el contexto de la asignación dinámica ocurre cuando se reserva espacio en memoria (con funciones como `malloc` en C o `new` en C++) pero no se libera adecuadamente utilizando `free` o `delete` cuando ya no es necesario. Como resultado, la memoria asignada permanece ocupada incluso después de que la variable que la referencia ya no está en uso, lo que lleva a una pérdida gradual de recursos y podría causar problemas de rendimiento o agotamiento de memoria.