Estrutura de dados

Edson Orivaldo Lessa Junior

Manipulação de ponteiros

- A manipulação de ponteiros correta evita erros de acesso à memória
- Estes erros podem levar a comportamentos indefinidos e falhas graves no programa.

Acesso a áreas de memória inválidas

- Se um ponteiro não for corretamente inicializado ou manipulado, ele pode apontar para áreas de memória inválidas, resultando em leitura ou escrita em locais não autorizados.
- Isso pode corromper dados importantes ou causar falhas de segmentação.

Vazamentos de memória

- Uma má manipulação de ponteiros pode resultar em vazamentos de memória, onde a memória alocada dinamicamente não é liberada adequadamente.
- Podendo levar a um consumo excessivo de memória e a problemas de desempenho no programa.

Corrupção de dados

- Se ponteiros forem usados de forma incorreta para acessar e modificar dados, pode ocorrer corrupção de dados em variáveis ou estruturas importantes.
- Podendo levar a resultados inesperados e bugs difíceis de rastrear.

Segurança do programa

- Erros na manipulação de ponteiros podem ser explorados por invasores para realizar ataques de segurança, como injeção de código malicioso ou exploração de vulnerabilidades de buffer overflow.
- Entender como os ponteiros funcionam ajuda a fortalecer a segurança do programa.

Desempenho do programa

- Uma manipulação eficiente de ponteiros pode melhorar significativamente o desempenho do programa, permitindo o acesso direto à memória e evitando operações desnecessárias de cópia de dados.
- Erros na manipulação de ponteiros podem resultar em um desempenho ruim do programa.

Debugging e manutenção do código

- Quando ocorrem erros relacionados a ponteiros, pode ser desafiador identificar a causa raiz e corrigir o problema.
- Um bom entendimento da manipulação de ponteiros facilita o processo de debugging e manutenção do código, tornando-o mais robusto e confiável.

Exemplo ponteiro errado

- Exemplo em código
- https://github.com/edson-lessa-jr/unicesumar-aula2-estrutura-de-dados

Alocação Dinâmica na Memória

O que é alocação dinâmica de memória

- A alocação dinâmica de memória é um conceito em programação que se refere à capacidade de um programa reservar e liberar blocos de memória conforme necessário durante a execução do programa
- A memória é alocada em tempo de compilação e permanece fixa durante toda a execução do programa

Quais benefícios da alocação dinâmica de memória

- A alocação dinâmica de memória é útil em situações em que o tamanho ou a quantidade de memória necessária não é conhecida em tempo de compilação, mas sim em tempo de execução
- Flexibilidade:
  - Alocar apenas a quantidade necessária de memória em tempo de execução. Isso é particularmente útil quando o tamanho dos dados a serem manipulados não é conhecido antecipadamente.

Quais benefícios da alocação dinâmica de memória

- Eficiência:
  - Uso mais eficiente da memória. Em vez de alocar uma quantidade fixa de memória para uma estrutura de dados no início do programa, a memória pode ser alocada conforme necessário, reduzindo assim o desperdício de memória.
- Gerenciamento de Memória Grande:
  - Em programas que lidam com grandes conjuntos de dados ou operam em sistemas com recursos limitados de memória, a alocação dinâmica de memória permite um gerenciamento mais eficaz da memória.
  - Por exemplo, em programas que processam grandes arquivos de dados ou imagens, a alocação dinâmica de memória pode ser usada para alocar apenas a quantidade necessária de memória conforme o processamento avança.
- Reutilização de Memória:
  - Reutilização eficiente da memória.
  - Por exemplo, quando um bloco de memória não é mais necessário, ele pode ser liberado e reutilizado por outras partes do programa, evitando assim a fragmentação da memória

Como usar

- Em linguagens como C e C++, a alocação dinâmica de memória é realizada usando funções como malloc(), calloc(), realloc() e free()
- **malloc():** Esta função aloca um bloco de memória especificado pelo número de bytes e retorna um ponteiro para o início desse bloco de memória.

```
int *ptr = (int *)malloc(10 * sizeof(int));
```

Como usar

- **calloc():** Esta função aloca um bloco de memória para um array de elementos, inicializa todos os bits de memória com zero e retorna um ponteiro para o início desse bloco de memória.

- **realloc():** Esta função altera o tamanho de um bloco de memória previamente alocado. Ela pode ser usada para aumentar ou diminuir o tamanho do bloco de memória e retorna um ponteiro para o bloco de memória realocado.

```
int *ptr = (int *)malloc(10 * sizeof(int));
ptr = (int *)realloc(ptr, 20 * sizeof(int));
```

Como usar

- **free():** Esta função libera um bloco de memória previamente alocado de volta para o sistema. Ela deve ser usada após o término do uso de um bloco de memória alocado dinamicamente para evitar vazamentos de memória.

```
free(ptr);
```

Exemplos

- https://github.com/edson-lessa-jr/unicesumar-aula3-estrutura-de-dados

Criando Vetores Dinâmico

Vetores dinâmicos

- Vetores dinâmicos ou arrays dinâmicos, são estruturas de dados em linguagens de programação que permitem alocar memória de forma dinâmica para armazenar uma coleção de elementos do mesmo tipo.
- Em linguagens como C, a alocação dinâmica de vetores é realizada utilizando funções como `malloc` para reservar espaço na memória conforme necessário.
- A principal característica dos vetores dinâmicos é a capacidade de ajustar seu tamanho durante a execução do programa, ao contrário dos vetores estáticos, cujo tamanho é definido em tempo de compilação e permanece fixo.

Vetores dinâmicos

- Proporciona flexibilidade ao programador para lidar com situações em que o tamanho do vetor pode variar ou não é conhecido antecipadamente.
- Ao criar um vetor dinâmico pode-se alocar espaço na memória para armazenar um número específico de elementos de um determinado tipo.
    - Por exemplo: um vetor dinâmico de inteiros pode ser criado alocando memória para um número variável de elementos inteiros.

Estruturas de dados que podem crescer ou diminuir em tamanho durante a execução de um programa

Vetores dinâmicos

- São úteis em situações em que o tamanho dos dados a serem armazenados não é conhecido antecipadamente ou pode variar durante a execução do programa.
- Permitem que o programa utilize a memória de forma mais eficiente, alocando apenas a quantidade necessária de memória conforme necessário.
- Oferecem flexibilidade para lidar com conjuntos de dados de tamanho variável, evitando o desperdício de memória associado aos arrays estáticos de tamanho fixo.

Quando usar

- Em C, os vetores dinâmicos são implementados com frequência quando há necessidade de lidar com conjuntos de dados cujo tamanho não é conhecido em tempo de compilação ou pode variar durante a execução do programa.
- Leitura de Dados de Tamanho Variável:
  - Quando os dados são lidos de uma entrada externa, como um arquivo ou entrada do usuário, e o tamanho dos dados não é conhecido antecipadamente, vetores dinâmicos podem ser usados para armazenar esses dados à medida que são lidos.

Quando usar

- Processamento de Dados em Tempo Real:
  - Em programas que lidam com aquisição de dados em tempo real, como processamento de sinais ou coleta de dados de sensores, o tamanho dos dados pode variar dinamicamente à medida que novos dados são recebidos. Vetores dinâmicos podem ser usados para armazenar esses dados à medida que são recebidos e processados.
- Implementação de Estruturas de Dados Dinâmicas:
  - Vetores dinâmicos são frequentemente usados na implementação de estruturas de dados dinâmicas, como listas encadeadas, filas e pilhas. Eles oferecem uma maneira eficiente de armazenar e manipular conjuntos de dados que podem crescer ou diminuir em tamanho durante a execução do programa.

Quando usar

- Gerenciamento de Memória Eficiente:
  - Em programas que precisam alocar e liberar memória dinamicamente para evitar o desperdício de memória associado aos arrays estáticos de tamanho fixo, vetores dinâmicos são uma solução eficiente. Eles permitem que o programa aloque apenas a quantidade necessária de memória conforme necessário.
- Implementação de Algoritmos Dinâmicos:
  - Em algoritmos que exigem uma quantidade variável de espaço de trabalho, como algoritmos de pesquisa, ordenação ou processamento de grafos, vetores dinâmicos podem ser usados para armazenar temporariamente dados durante a execução do algoritmo.

Como usar

- Em C, os vetores dinâmicos são implementados usando as funções como malloc(), calloc(), realloc() e free().
- Inicialmente, um vetor dinâmico é alocado com um tamanho inicial, e conforme mais elementos são adicionados, o vetor pode ser realocado com um tamanho maior usando realloc().
- Da mesma forma, se elementos são removidos do vetor, ele pode ser realocado com um tamanho menor.

Exemplo

- https://github.com/edson-lessa-jr/unicesumar-aula3-estrutura-de-dados

BONS ESTUDOS