



TIC – Hub
Programa Residência em TIC 12

C A P A C I T A Ç ã O

CURSO

Estruturas de dados

Capítulo 4

Ponteiros, registros e listas encadeadas

Prof. Allberson Dantas

EXECUTORES



PARCEIROS



COORDENAÇÃO PPI



INICIATIVA



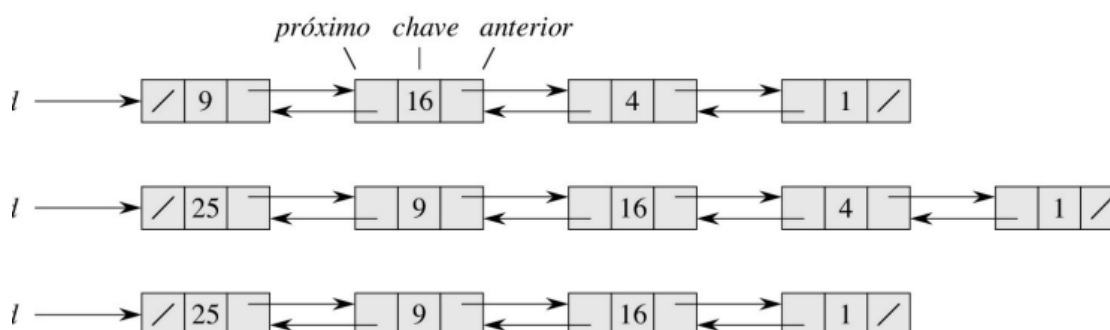
Iniciativa do Ministério da Ciência, Tecnologia e Inovações e Softex no âmbito do Programa MCTI FUTURO. É um Projeto apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, conforme disposto no Art. 7º da Portaria MCTI Nº 5.275, de 5 de novembro de 2021. Este projeto é apoiado pelo Ministério da Ciência, Tecnologia e Inovações, com recursos da Lei nº 8.248, de 23 de outubro de 1991, no âmbito do PPI-Softex, coordenado pela Softex e publicado no Projeto de Residência em TIC 12. Leia o termo de consentimento para tratamento de dados pessoais.

Olá, seja bem-vindo(a) a mais uma aula.

Desta vez você aprenderá **ponteiros**, ferramentas poderosas que exploram os meandros da memória e elevam seus programas a um novo patamar de eficiência.

Para entender melhor, imagine um mapa do tesouro, que guarda o segredo de riquezas inigualáveis. Para encontrar o tesouro, precisamos de um guia experiente: os ponteiros!

Na programação, os ponteiros funcionam como setas que apontam para endereços específicos na memória. Ao declarar um ponteiro, definimos uma variável que armazena o endereço de outra variável. Isso permite que o ponteiro acesse o valor da variável original de forma direta, sem a necessidade de cópias desnecessárias:



Fonte: Cormen et. al., 2009

Observamos bem essa representação na imagem acima, que é de listas encadeadas; as setas funcionam como ponteiros.

Para dominar a arte dos ponteiros, é fundamental compreender os conceitos básicos que definem seu funcionamento:

- **Variável:** uma variável é uma entidade na programação que armazena um valor em um local específico da memória;
- **Endereço:** cada variável possui um endereço único na memória, que a identifica e permite seu acesso;
- **Ponteiro:** um ponteiro é uma variável especial que armazena o endereço de outra variável;
- **Desreferenciação:** o operador * (asterisco) é utilizado para acessar o valor da variável apontada por um ponteiro;
- **Operador de endereço:** o operador & (e comercial) é utilizado para obter o endereço de uma variável.

Os ponteiros podem ser utilizados para diversas tarefas, como:

- **Acessar e manipular variáveis de forma direta:** os ponteiros leem, escrevem e modificam o valor de variáveis na memória, sem utilizar mecanismos indiretos como referências;
- **Passar parâmetros por referência:** ao passar ponteiros como parâmetros para funções, é possível modificar as variáveis originais dentro da função, otimizando a passagem de dados e evitando cópias desnecessárias;
- **Retornar valores por meio de ponteiros:** funções podem retornar ponteiros como valor de retorno, permitindo que o programador acesse e manipule os dados diretamente na memória principal.

Benefícios da utilização de ponteiros

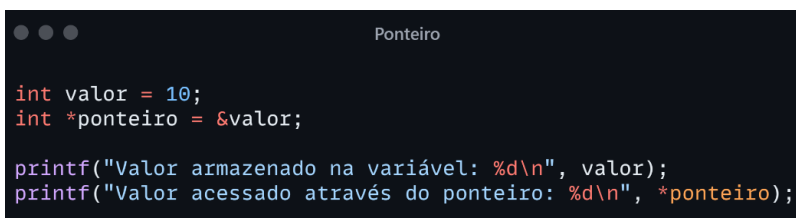
Os ponteiros oferecem diversos benefícios que os tornam ferramentas indispensáveis para programadores experientes:

- **Eficiência:** acessam diretamente a memória, otimizando operações que manipulam grandes quantidades de dados, como arrays e matrizes;
- **Flexibilidade:** possibilita a criação de estruturas de dados complexas e dinâmicas, como listas encadeadas e árvores, expandindo as possibilidades de armazenamento e manipulação de informações;
- **Controle detalhado:** manipula diretamente a memória, abrindo novas possibilidades de programação e personalização do comportamento do programa.

Exemplo prático: acessando valores através de ponteiros

Até aqui, todos os códigos foram implementados na linguagem Java, porém o exemplo abaixo foi implementado na **linguagem C**. **Por qual motivo?** A linguagem Java não possui suporte nativo para ponteiros como a linguagem C.

Em Java, a manipulação da memória é gerenciada automaticamente pela máquina virtual Java (JVM), o que contribui para a segurança e simplicidade da linguagem.



```
int valor = 10;
int *ponteiro = &valor;

printf("Valor armazenado na variável: %d\n", valor);
printf("Valor acessado através do ponteiro: %d\n", *ponteiro);
```

Neste exemplo, demonstramos como um ponteiro é utilizado para acessar o valor de uma variável de forma direta. O ponteiro (chamado ponteiro) armazena o endereço da variável valor, permitindo que o programa acesse o valor 10 armazenado na variável original através da desreferenciação (*ponteiro).

Alocação Dinâmica de Memória

Um dos recursos mais poderosos dos ponteiros é a possibilidade de realizar **alocação dinâmica de memória**. Isso significa que a memória pode ser alocada durante a execução do programa, de acordo com a necessidade, e não apenas na declaração inicial das variáveis. Para realizar a alocação dinâmica de memória, utilizamos funções como `malloc` e `free` da biblioteca padrão C.

A função `malloc` permite alocar memória durante a execução do programa, retornando o endereço do bloco de memória alocado.

É fundamental liberar a memória alocada dinamicamente, utilizando a função `free` quando ela não for mais necessária. O não-cumprimento dessa regra pode levar a vazamentos de memória, comprometendo o desempenho e a estabilidade do programa.

Considerações importantes

O uso de ponteiros é um recurso poderoso, mas requer cautela e responsabilidade por parte do programador. É importante estar atento às seguintes considerações:

- **Segurança:** o uso incorreto de ponteiros pode levar a erros de segmentação e outros problemas graves. Acessar endereços de memória inválidos ou tentar liberar memória já liberada pode causar instabilidade no programa;
- **Gerenciamento de memória:** o gerenciamento manual de memória utilizando `malloc` e `free` exige cuidado para evitar vazamentos de memória. Certifique-se de liberar toda a memória alocada dinamicamente quando ela não for mais necessária;

- **Complexidade:** ponteiros exigem um bom entendimento de como a memória é gerenciada pelo computador. É fundamental compreender conceitos como endereçamento de memória, alocação dinâmica e manipulações de baixo nível para utilizar ponteiros de forma segura e eficiente.

Ponteiros são ferramentas poderosas que expandem as possibilidades da programação, permitindo acesso direto à memória, criação de estruturas de dados complexas e otimização do desempenho. Dominar os ponteiros requer dedicação e prática, mas o conhecimento adquirido abre portas para o desenvolvimento de programas mais eficientes, flexíveis e com controle granular da memória.

Nesta aula, exploramos os conceitos básicos de ponteiros. Continuem estudando e praticando, pois os ponteiros são elementos fundamentais em muitas linguagens!