

Estrutura de Dados

prof: PAULO ROBERTO FREITAS RODRIGUES
FAMETRO ADS 4º

Aluna Adila Zaira Oliveira

Estrutura de dados é uma forma de organizar e armazenar dados de maneira eficiente, facilitando a manipulação, acesso e operações sobre esses dados. Em outras palavras, é um conjunto de técnicas para representar informações de forma organizada na memória do computador.

Tipos de Dados

Tipos básicos (primitivos): inteiro, real, lógico e caractere;

- Tipos de estruturados (construídos): array
- vetores unidimensional
- matrizes multidimensional

Categoria	Tipo	Tamanho
Inteiro	byte	8 bits
Inteiro	short	16 bits
Inteiro	int	32 bits
Inteiro	long	64 bits
Ponto Flutuante	float	32 bits
Ponto Flutuante	double	64 bits
Caracter	char	16 bits
Lógico	boolean	true / false

Estrutura Condicional

Uma estrutura condicional permite que o programa faça a escolha do que executar, de acordo com uma condição.

- if
- if else
- switch

Estrutura de Repetição

As estruturas de repetição permitem executar mais de uma vez um mesmo trecho de código. Trata-se de uma forma de executar blocos de comandos somente sob determinadas condições, mas com a opção de repetir o mesmo bloco quantas vezes for necessário.

- for
- while
- do while
- loop

Estruturas de repetição (ou laços) são fundamentais na programação para executar um bloco de código múltiplas vezes. As principais estruturas de repetição em muitas linguagens são:

1. for:

Usado quando o número de iterações é conhecido ou pode ser determinado antes da execução. Ideal para percorrer listas, arrays, ou realizar uma ação um número fixo de vezes.

2. while:

Usado quando não se sabe quantas vezes o código será repetido, mas se tem uma condição que deve ser verdadeira para continuar. Continua executando enquanto a condição for verdadeira.

3. do...while :

(em algumas linguagens, como C ou JavaScript): Semelhante ao while, mas garante que o código seja executado pelo menos uma vez, já que a condição é verificada ao final do laço.

Estrutura de Dados

prof: PAULO ROBERTO FREITAS RODRIGUES
FAMETRO ADS 4º

Aluna Adila Zaira Oliveira

Como decidir qual usar?

1. Se o número de repetições é conhecido ou pode ser facilmente determinado no início, o laço for é geralmente a melhor escolha.
2. Se o número de iterações não é conhecido, mas há uma condição de término, como esperar por uma entrada ou por uma condição externa, o laço while é mais adequado.
3. Se você precisa que o código seja executado pelo menos uma vez e só depois verificar a condição, o laço do...while (onde disponível) é o mais apropriado.

Variável Composta Homogênea: É uma variável que armazena valores de um único tipo de dados. Isso significa que todos os elementos contidos nessa variável são do mesmo tipo.

Variável Composta Heterogênea: É uma variável que armazena valores de diferentes tipos de dados. Isso significa que os elementos contidos nessa variável podem ser de tipos distintos.

Funções

- Um programa em C é um conjunto de funções definidas pelo programador;
- Funções que utilizarão outras funções definidas pelo programador;
- As funções oferecidas pelo sistema são chamadas funções de biblioteca ou funções pré-definidas;

- Todo programa em C deve conter uma função identificada por main (principal), com lista de parâmetros vazia e tipo de dado não obrigatório;

Uma função definida pelo programador (ou função definida pelo usuário) é uma função que é escrita pelo próprio programador como parte do código-fonte do programa. Por outro lado, uma função de biblioteca é uma função que já foi implementada por terceiros e disponibilizada em uma biblioteca padrão ou externa para uso em programas.

Cada função, além de ter acesso às variáveis do programa que o chamou (são as variáveis globais), pode ter suas próprias variáveis (são variáveis locais), que existem apenas durante sua chamada.

- Declaração de uma função em C:
- `<tipo-de-dado> <nome-da-função> (<sequência-de-declarções-de-parâmetros>) {`

/comandos

Exemplo:

```
//função principal do C
int main () {
//comandos return 0;
```

Estrutura de Dados

prof: PAULO ROBERTO FREITAS
RODRIGUES FAMETRO ADS 4º

Variável Global:

- Uma variável global é declarada fora de qualquer função em um programa e, portanto, está disponível em todo o programa, em qualquer função.
- Essas variáveis podem ser acessadas e modificadas de qualquer lugar do código.
- São inicializadas apenas uma vez, no início do programa, e mantêm seu valor durante toda a execução do programa.
- Variáveis globais podem ser úteis para armazenar valores que precisam ser acessados por várias funções em um programa.

Variável Local:

- Uma variável local é declarada dentro de uma função e, portanto, só está acessível dentro dessa função.
- Elas existem apenas durante a execução da função em que foram declaradas e são destruídas quando a função é concluída.
- Variáveis locais têm escopo limitado ao bloco em que são declaradas.
- Cada chamada de função cria uma nova instância das variáveis locais

Vetores

Um vetor é uma variável composta unidimensional formada por uma sequência de variáveis, todas do mesmo tipo, com o mesmo nome e alocadas sequencialmente na memória. Uma vez que as variáveis têm o mesmo nome, o que as distingue é um índice, que referencia sua localização dentro da estrutura.

Declaração de um vetor

A declaração de um vetor é feita da seguinte maneira:

tipo nome[tamanho]

Onde:

- tipo é o tipo de dado que será armazenado no vetor;
- tamanho é quantidade de variáveis que vão compor o vetor;
- nome é o nome da variável do tipo vetor

Matrizes

Uma matriz é uma coleção de variáveis de mesmo tipo, com mesmo nome e alocadas sequencialmente na memória. Uma vez tendo o mesmo nome, o que as distingue são índices que referenciam sua localização dentro da estrutura.

As matrizes podem ser tanto unidimensionais (vetores) como multidimensionais.

Como já trabalhamos com vetores, daremos atenção as matrizes com duas ou mais dimensões.

Matriz bidimensional

Declaração

A declaração de uma matriz bidimensional é feita da seguinte maneira:

tipo nome[linha][coluna];

Onde:

- tipo é o tipo de dado que será armazenado na matriz;
- linha e coluna correspondem a quantidade de linhas e colunas da matriz;
- nome é o nome da variável do tipo matriz.

Estrutura de Dados

prof: PAULO ROBERTO FREITAS RODRIGUES
FAMETRO ADS 4º

Aluna Adila Zaira Oliveira

tipos de estruturas:

Quando falamos em estrutura de dados, é importante saber que existem diversos tipos de estruturas utilizados na apresentação.

Abaixo, os mais conhecidos e usados:

- Vetor (Array): uma estrutura linear e estática composta por um número fixo de elementos. É recomendado quando os dados armazenados não sofrerão mudanças significativas ao longo do tempo.
- Lista: uma estrutura linear e dinâmica, constituída por nós que apontam para o elemento seguinte (exceto o último).
- Árvore: nessa estrutura, cada elemento possui pelo menos um outro elemento associado a ele, formando uma estrutura hierárquica.
- Fila: baseia-se no princípio FIFO (First-In, First-Out), ou seja, os elementos inseridos primeiro serão os primeiros a serem removidos.
- Pilha: segue o princípio LIFO (Last-In, First-Out), em que os elementos inseridos por último serão removidos primeiro.
-

Estrutura de Dados

prof: PAULO ROBERTO FREITAS
RODRIGUES FAMETRO ADS 4º

Ponteiros

Em programação, um ponteiro é uma variável que armazena o endereço de outra variável. Isso significa que, ao invés de guardar um valor diretamente (como um número ou uma letra), o ponteiro guarda a localização de onde esse valor está na memória.

- O ponteiro é um tipo de dado como int, char ou float.
- Um ponteiro aponta para algo. Em programação, temos as variáveis armazenadas na memória, e um ponteiro aponta para um endereço de memória.
- Imagine as variáveis como documentos, a memória do computador pastas para guardar os documentos, e o ponteiro como atalhos para as pastas.

Declaração de Ponteiros

Para declarar um ponteiro, especificamos o tipo da variável para a qual ele aponta e seu nome precedido por asterisco. Pode-se ter um ponteiro para qualquer tipo de variáveis possível em C.

Não confunda o uso de "*" com o operador de multiplicação.

• Tipo *variável;

- `int *a;` /*ponteiro para inteiro*/
- `char *a;` /*ponteiro para char*/
- `float *a;` /*ponteiro para float*/

Como Usar Ponteiros

1. Declaração:

```
int *p; // Declara um ponteiro para int
```

2. Inicialização:

Depois, você pode inicializar o ponteiro para apontar para uma variável.

```
int x = 10; // Uma variável do tipo int
p = &x; // O ponteiro p agora aponta para x
```

- Aqui, `&x` é o operador "endereço de", que fornece o endereço de memória de x.

3. Uso: Você pode acessar o valor para o qual o ponteiro aponta usando o operador *, chamado "desreferência".

```
printf("%d\n", *p); // Imprime o valor de x, que é 10
```

O `*p` acessa o valor na memória para o qual p está apontando.

Aqui está um exemplo completo em C:

```
#include <stdio.h>
```

```
int main() {
```

```
    int x = 10; // Declara uma variável int
    int *p = &x; // Declara um ponteiro p e faz
    ele apontar para x
```

```
    printf("Valor de x: %d\n", x); // Imprime o
    valor de x
```

```
    printf("Valor apontado por p: %d\n", *p); // Imprime o valor de x usando o ponteiro
    return 0;
}
```

Neste código:

- `x` é uma variável que guarda o valor 10.
- `p` é um ponteiro que guarda o endereço de `x`.
- `*p` nos dá o valor de `x` através do ponteiro.

Pilha

O que é uma Pilha?

Uma pilha é uma estrutura de dados que segue a regra LIFO (Last In, First Out), ou seja, o último elemento a ser inserido é o primeiro a ser removido. Pense em uma pilha de pratos: você só pode pegar o prato que está no topo.

Operações Básicas

1. Push: Adiciona um elemento ao topo da pilha.
2. Pop: Remove e retorna o elemento do topo da pilha.
3. Peek (ou Top): Retorna o elemento do topo da pilha sem removê-lo.
4. IsEmpty: Verifica se a pilha está vazia.

Estrutura de Dados

prof: PAULO ROBERTO FREITAS
RODRIGUES FAMETRO ADS 4º

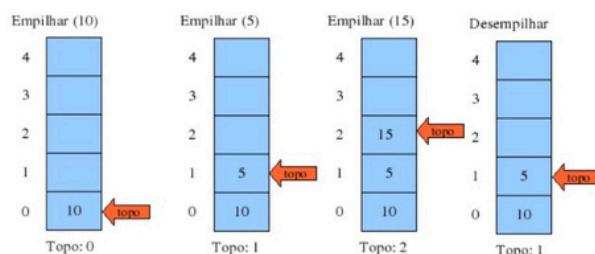
As pilhas encontram inúmeras aplicações em programação e desenvolvimento de algoritmos, como por exemplo:

- Avaliação de Expressões e Parsing Sintático
- Gerenciamento de Memória em tempo de compilação
- Controle de Navegação em browsers
- Endereçamento de instruções em microprocessadores
- Análise de expressões aritméticas

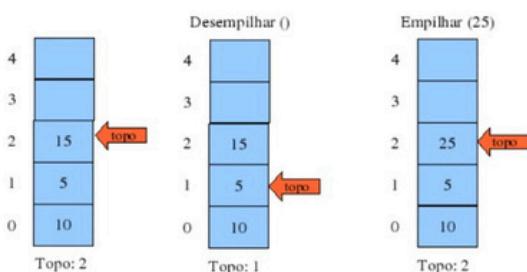
A implementação de pilhas pode ser realizada através de vetor (alocação do espaço de memória para os elementos é contígua) ou através de listas encadeadas.

Numa pilha, a manipulação dos elementos é realizada em apenas uma das extremidades, chamada de topo, em oposição a outra extremidade, chamada de base.

Ex: Supondo uma pilha com capacidade para 5 elementos (5 nós).



Na realidade a remoção de um elemento da pilha é realizada apenas alterando-se a informação da posição do topo.



Uma pilha bem formada é aquela em que as operações respeitam as regras da estrutura e onde o estado final da pilha reflete uma sequência correta e válida de operações.

Em uma expressão com esses símbolos, a ideia é garantir que:

1. Cada abertura de um símbolo ((), [], {}) tenha um fechamento correspondente ((), [], {}).
2. Os símbolos sejam fechados na ordem correta: o último a ser aberto deve ser o primeiro a ser fechado.