

Apresentação

Para o profissional da área da computação, é importante conhecer a forma como os dados são armazenados, organizados e manipulados, pois essa atividade é uma das principais desenvolvidas pelos sistemas computacionais. Nesse sentido, a estrutura de dados é um elemento que vem, principalmente na linguagem de programação C, para auxiliar na representação e na abstração de estruturas mais complexas, executando operações de armazenamento e busca de dados na memória, de maneira mais sofisticada e robusta.

Nesta Unidade de Aprendizagem, você vai estudar a estrutura de dados, dados heterogêneos e homogêneos, e a aplicação de dados homogêneos, como vetores e matrizes, e ponteiros.

Bons estudos.

Ao final desta Unidade de Aprendizagem, você deve apresentar os seguintes aprendizados:

- Identificar uma estrutura de dados.
- Reconhecer dados heterogêneos e homogêneos.
- Aplicar dados homogêneos (vetores e matrizes) e ponteiros.

Desafio

Os vetores e as matrizes são os exemplos mais comuns de estruturas de dados em C que são formadas por dados homogêneos, ou seja, são formadas por somente um tipo de dados.

Para resolver a situação a seguir, considere a aplicação de dados homogêneos (vetores e matrizes) e ponteiros.

Você está com a lista de preços de uma papelaria, que contém os seguintes materiais e preços normal e com desconto:

	PREÇO[0]	PREÇO[1]
MATERIAL[0]	15,00	12,50
MATERIAL[1]	13,00	7,50
MATERIAL[2]	100,00	97,00

Você deverá elaborar um código em linguagem C para apresentar um vetor que contenha a média de preços por produto. O vetor deve ser apresentado de forma semelhante a este:

13,75	10,25	98,50
-------	-------	-------

```
#include <stdio.h>

int main() {
    // Definição dos preços normais e com desconto para cada material
    float precos[3][2] = {
        {15.00, 12.50},
        {13.00, 7.50},
        {100.00, 97.00}
    };

    // Vetor para armazenar a média de preços por produto
    float media_precos[3] = {0}; // Inicializado com 0

    // Cálculo da média de preços por produto
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
            media_precos[i] += precos[i][j]; // Soma dos preços normais e com desconto
        }
        media_precos[i] /= 2; // Divide a soma pelo número de preços (2)
    }

    // Apresentação dos resultados
    printf("Média de preços por produto:\n");
    for (int i = 0; i < 3; i++) {
        printf("%.2f\n", media_precos[i]);
    }
    printf("\n");
}
```

DECLARAÇÃO DE ESTRUTURA DE DADOS EM C

```
struct boletim (
    char nome_do_aluno(50);
    char nome_disciplina(50);
    int qtd_faltas;
    float nota1;
    float nota2;
    float nota_exam;
);
```

Infográfico

As estruturas de dados normalmente envolvem a identificação e o desenvolvimento de entidades e operações úteis, e determinam o tipo de problema que pode ser solucionado por meio da sua utilização. Elas também determinam a representação de entidades abstratas e a implementação de operações abstratas que podem ser utilizadas com representações concretas.

Veja, no Infográfico a seguir, a ilustração sobre a estrutura de dados.

O QUE É UMA ESTRUTURA DE DADOS?

SINTAXE DA DECLARAÇÃO:

```
struct nome_da_estrutura {  
    tipo_do_campo1 nome_do_campo1;  
    tipo_do_campo2 nome_do_campo2;  
    tipo_do_campo3 nome_do_campo3;  
    ...  
};
```

Serve para auxiliar o armazenamento, no computador, de dados que são vistos na vida real e, por isso, precisam de abstração.

É formada por elementos, membros ou campos.

Agrupa e manipula dados dentro de uma entidade identificada por meio de um único nome de variável.

Envolve o armazenamento de dados organizados na memória de forma mais sofisticada do que utilizando variáveis simples.

Permite executar operações e fazer manipulação de dados, indicando como os dados são representados e a forma possível de manipulá-los.

Faz referência aos dados utilizando índices que servem para otimizar a localização de um registro quando é feita uma consulta que envolve vários dados.



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Conteúdo do Livro

A área da Ciência da Computação trata basicamente sobre a maneira como os dados são armazenados, organizados e manipulados, logo, como esse tipo de operação faz parte do cotidiano dos profissionais desenvolvedores de sistemas, torna-se importante entender como é executado. Nesse sentido, a estrutura de dados é um elemento que vem, principalmente na linguagem de programação C, para auxiliar na representação e na abstração de estruturas mais complexas, executando operações de armazenamento e busca de dados na memória, de maneira mais sofisticada e robusta.

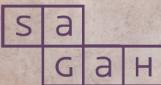
As estruturas de dados mais conhecidas são os vetores e as matrizes, que são estruturas unidimensionais, no caso dos primeiros, e multidimensionais, no caso das últimas. Eles introduzem a noção de índice para otimizar a localização de um registro quando uma consulta envolve vários conjuntos de dados.

No capítulo **Estrutura de dados**, do livro *Estrutura de dados*, você vai estudar sobre estrutura de dados, dados heterogêneos e homogêneos, e sobre a aplicação de dados homogêneos, como vetores e matrizes, e ponteiros.

Boa leitura.

ESTRUTURA DE DADOS

Jeanine dos Santos Barreto



SOLUÇÕES
EDUCACIONAIS
INTEGRADAS

Estrutura de dados

Objetivos de aprendizagem

Ao final deste capítulo, você deve apresentar os seguintes aprendizados:

- Identificar uma estrutura de dados.
- Reconhecer dados heterogêneos e homogêneos.
- Aplicar dados homogêneos (vetores e matrizes) e ponteiros.

Introdução

Para o profissional da área da Ciência da Computação, é importante conhecer a forma como os dados são armazenados, organizados e manipulados, pois essa atividade é uma das principais que são desenvolvidas pelos sistemas computacionais.

Nesse sentido, a estrutura de dados é um elemento que vem, principalmente na linguagem de programação *C*, para auxiliar na representação e na abstração de estruturas mais complexas, executando operações de armazenamento e busca de dados na memória, de maneira mais sofisticada e robusta.

Neste capítulo, você vai estudar sobre estrutura de dados e dados heterogêneos e homogêneos e sobre a aplicação de dados homogêneos, como vetores e matrizes, e ponteiros.

A estrutura de dados

A área da Ciência da Computação trata diretamente com o armazenamento, a organização, a manipulação e a utilização de dados e informações, por isso, é importante entender como isso acontece. Estudar estruturas de dados, principalmente na linguagem *C*, envolve (TENENBAUM; LANGSAM; AUGENSTEIN, 1995):

- A identificação e o desenvolvimento de entidades e operações que sejam úteis e determinem quais tipos de problemas podem ser solucionados pela sua utilização.
- A determinação de representações para entidades abstratas.
- A implementação de operações abstratas que possam ser utilizadas sobre representações concretas.



Fique atento

Um dado é algo normalmente quantificável, o qual não tem um significado relevante de maneira isolada.

O dado é considerado como o fundamento da informação. Conhecendo-se somente o dado, dificilmente é possível chegar a um entendimento sobre o assunto tratado, tomar decisões ou chegar a conclusões.

Já a informação consiste na ordenação e na organização dos dados, de maneira que se torne possível compreender o seu significado, ou entendê-la dentro de um contexto. Em outras palavras, a informação é o fruto do processamento dos dados, depois que estes são analisados, interpretados de uma maneira pré-definida e qualificados.

O nosso conhecimento é oriundo da consolidação dos dados em forma de informação, logo, quanto mais um indivíduo se distancia dos dados, maior é a sua capacidade de abstrair a realidade.

Quando são determinadas representações para entidades abstratas, é muito importante que sejam observadas as especificações dos recursos disponíveis para construir essas representações. Foi dito que essas características fazem parte da estrutura de dados na linguagem *C*, pois muitas outras linguagens já têm o conceito de estrutura de dados incorporados na sua estrutura. A sua utilização pode até ser mais facilitada, mas nem sempre é eficiente.

A eficiência de uma estrutura de dados envolve, normalmente, variáveis, como o tempo e o espaço. Se uma aplicação precisa manipular estruturas de dados de forma muito intensa e repetida, a velocidade com que essas manipulações são feitas será decisiva. Da mesma forma, uma aplicação que precisa manipular uma grande quantidade de estruturas de dados, e tem um código gigantesco para fazer a representação dessas estruturas, não será eficiente (LAUREANO, 2008).

A utilização de uma estrutura de dados serve, via de regra, para que os dados que são vistos na vida real possam ser armazenados pelo computador, daí o seu grau de abstração. As estruturas de dados envolvem o armazenamento de dados organizados na memória de uma maneira mais sofisticada do que pela utilização de variáveis básicas usadas em algoritmos simples. Quando se deseja representar dados de forma direta, a referência a eles é feita normalmente por meio de variáveis, mas quando se utiliza uma estrutura de dados, a referência aos dados é feita normalmente por índices.



Fique atento

Quando o assunto é estrutura de dados, um índice é uma referência que se associa a uma chave, a qual serve para otimizar a localização de um registro, quando é feita uma consulta que envolve vários dados.

Uma estrutura de dados é um elemento que permite executar operações e fazer a manipulação de dados. Ela serve para indicar como os dados são representados e como se torna possível a sua manipulação. Um exemplo muito comum para entender a estrutura de dados é a representação de um baralho de cartas:

- A representação das cartas pode ser feita por dois valores do tipo caractere, que são o naipe e o valor da carta.
- As operações que podem ser feitas com as cartas são:
 - a) embaralhar todas as cartas;
 - b) comprar uma carta do topo do baralho;
 - c) retirar uma carta de um lugar qualquer do meio do baralho;
 - d) inserir uma carta na base do baralho.

Só é possível escolher o tipo de estrutura de dados que será utilizada na aplicação depois de analisar e entender todos os tipos de operações que serão possíveis de realizar com os dados, pois as estruturas de dados podem ser meios elegantes de organizá-los, mas uma escolha errada pode trazer desvantagens, principalmente de performance.

As estruturas de dados servem para agrupar e manipular objetos dentro de uma entidade que seja identificada por meio de um único nome de variável. Normalmente, a estrutura de dados é formada por variáveis de tipos diferentes, as quais são chamadas de membros, elementos ou campos da estrutura.

A declaração de uma estrutura de dados normalmente é feita pela sua nomeação e pela indicação dos campos que farão parte dela, como o tipo e o nome das variáveis. Algumas observações importantes sobre a declaração da estrutura de dados são:

- A chave que fecha a declaração da estrutura de dados (*struct*) deve ser seguida de um ponto-e-vírgula.
- Não é possível que mais de um campo tenha o mesmo nome.

Essa é a forma de declarar uma estrutura de dados na linguagem de programação *C*:

```
struct nome_da_estrutura {  
    tipo_do_campo1 nome_do_campo1;  
    tipo_do_campo2 nome_do_campo2;  
    tipo_do_campo3 nome_do_campo3;  
    ...  
};
```

Com base na declaração acima, este é um exemplo de declaração de estrutura de dados em *C*:

```
struct boletim {  
    char nome_do_aluno(50);  
    char nome_disciplina(50);  
    int qtd_faltas;  
    float nota1;  
    float nota2;  
    float nota_exame;  
};
```

Dados heterogêneos e homogêneos

Quando uma estrutura de dados armazena elementos de tipos diferentes, é dito que esta é uma estrutura de dados heterogênea. Normalmente, esse tipo de dado é chamado de registro, então, o registro é uma estrutura de dados que faz o agrupamento de dados de tipos diferentes entre si. Ele é formado por uma quantidade determinada de campos, os quais são itens individuais de dados e relacionados de forma lógica (LAUREANO, 2008).

Em outras palavras, um registro é algo que agrupa elementos que não são do mesmo tipo de dado, mas que têm vínculo lógico. Os registros consistem em conjuntos de posições de memória, os quais são identificados pelo mesmo nome e são individualizados por meio de identificadores que se associam a cada conjunto de posições (Figura 1).

Funcionário		
Marcos Laureano		
27	12	1975
Informática		
R\$ 3.000,00		

Figura 1. Representação gráfica de um registro.

Fonte: Laureano (2008, p. 41).

Para referenciar um único campo dentro de um registro, é preciso utilizar essa sintaxe:

```
nome_da_estrutura[índice].nome_do_campo;
```

Por sua vez, uma estrutura de dados que armazena somente um tipo de dado é conhecida, então, como *dados homogêneos*. Os elementos que compõem uma estrutura de dados homogênea correspondem às posições de memória, as quais são identificadas por meio de um mesmo nome, individualizadas por índices que são todos dos mesmos tipos de dados. Os dados homogêneos são representados pelos vetores, ou estruturas de dados unidimensionais, e pelas matrizes, que são estruturas de dados bidimensionais (LAUREANO, 2008).

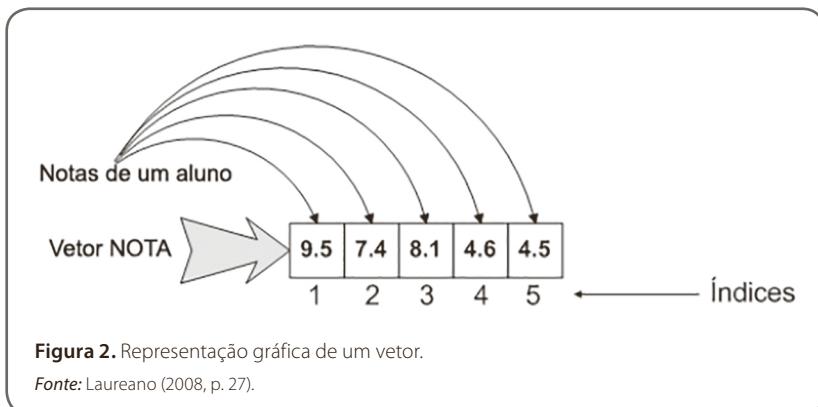
Vetores

Os vetores são estruturas de dados unidimensionais lineares e, por isso, precisam de um único índice para fazer o endereçamento e para percorrer toda a estrutura. Um vetor é uma estrutura de dados utilizada para fazer o armazenamento de uma lista de valores do mesmo tipo, ou seja, em uma mesma variável, vários valores de mesmo tipo são armazenados.

Quando um vetor é definido, é declarada uma quantidade fixa de posições que ele deve ter, ou seja, um vetor é um elemento que será dividido em várias posições que serão entendidas e reconhecidas pelo computador. Cada uma dessas posições vai ser responsável por armazenar um único valor do vetor e cada uma delas será reconhecida por um índice, ou endereço, e será por ele que essa posição será encontrada e poderá ser referenciada. Logo, apesar de essa estrutura ser de somente um tipo de dado, ela terá vários valores diferentes do mesmo tipo.

Uma estrutura de dados do tipo vetor tem as seguintes características (DEITEL; DEITEL, 2011):

- É alocado de forma estática, ou seja, no momento da sua declaração, deve-se conhecer o tamanho ou a quantidade de posições que ele terá.
- É uma estrutura de dados homogênea, formada por elementos com o mesmo tipo de dado.
- Cada posição do vetor contém somente um valor.
- Os tamanhos dos valores de cada posição, por serem do mesmo tipo, são iguais.
- Faz a alocação dos dados de forma sequencial (Figura 2).



A sintaxe para a definição do vetor acima seria:

tipo_de_dado nome_do_vetor[quantidade_de_posicoes_do_vetor]

Em C:

float nota[5];

O funcionamento de um vetor consiste em determinar a localização de todos os seus elementos, partindo do endereço do primeiro elemento, o que se torna possível porque todos estes ficam dispostos lado a lado e cada elemento tem um tamanho fixo. É importante lembrar que a primeira posição de um vetor será sempre conhecida por 0 e nunca por 1.

```
nota[0] = 9.5;
nota[1] = 7.4;
nota[2] = 8.1;
nota[3] = 4.6;
nota[4] = 4.5;
```

Matrizes

As matrizes são estruturas de dados bidimensionais, os quais precisam de dois índices para fazer o endereçamento e para percorrer toda a estrutura: um que servirá para referenciar a linha e outro que servirá para referenciar a coluna da matriz (TENENBAUM; LANGSAM; AUGENSTEIN, 1995).

Assim como os vetores, as matrizes têm algumas características:

- são alocadas de forma estática, ou seja, no momento da sua declaração, deve-se conhecer o tamanho que elas terão;
- são uma estrutura de dados homogênea, a qual é formada por elementos com o mesmo tipo de dados;
- cada posição da matriz contém somente um valor;
- os tamanhos dos valores de cada posição, por serem do mesmo tipo, são iguais;
- fazem a alocação dos dados de forma sequencial (Figura 3).

O diagrama mostra uma matriz 3x6 (3 linhas e 6 colunas) com os seguintes valores:

	1	2	3	4	5	6
1	M	A	R	C	O	S
2	N	A	S	S	E	R
3	D	O	N	A	L	D

As linhas são rotuladas com os números 1, 2 e 3, e uma seta aponta para cima, rotulada com "Linhas". As colunas são rotuladas com os números 1 a 6, e uma seta aponta para a direita, rotulada com "Colunas".

Figura 3. Representação gráfica de uma matriz.

Fonte: Laureano (2008, p. 31).

Uma matriz envolve dois ou mais vetores, os quais são definidos por um conjunto de elementos, ou seja, cada dimensão da matriz é formada por um vetor. A primeira dimensão de elementos é considerada o primeiro vetor, a segunda é considerada o segundo vetor, a terceira dimensão de elementos é considerada o terceiro vetor, e assim por diante.

A sintaxe para a definição da matriz acima seria:

tipo_de_dado

nome_da_matriz[quantidade_de_linhas][quantidade_de_colunas]

Em C:

```
char letras[3][6];
```

A alocação de valores para os elementos da matriz pode ser feita linha por linha ou coluna por coluna. É importante lembrar, da mesma forma que os vetores, que a primeira posição de cada dimensão da matriz será sempre conhecida por 0 e nunca por 1.

```
letras[0][0] = "M";
letras[0][1] = "A";
letras[0][2] = "R";
letras[0][3] = "C";
letras[0][4] = "O";
letras[0][5] = "S";
letras[1][0] = "N";
letras[1][1] = "A";
letras[1][2] = "S";
letras[1][3] = "S";
letras[1][4] = "E";
letras[1][5] = "R";
letras[2][0] = "D";
letras[2][1] = "O";
letras[2][2] = "N";
letras[2][3] = "A";
letras[2][4] = "L";
letras[2][5] = "D";
```

Aplicação de dados homogêneos (vetores e matrizes) e ponteiros

Existem muitas operações que podem ser feitas com vetores e matrizes na linguagem de programação C e as mais básicas serão exibidas a seguir.

Para que se possa compreender a aplicação de um vetor, pode-se criar um arquivo em C com o nome *vetor.cpp*, cujo código será o seguinte:

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define elementos 10
6
7 main() {
8     int posicao, vetor[elementos], maior = 0;
9
10    for(posicao=0;posicao<elementos;posicao++){
11        vetor[posicao] = posicao;
12    }
13
14    printf("O vetor criado contem os seguintes elementos: \n\n|");
15    for(posicao=0; posicao<elementos; posicao++)

```

```

16        printf("%d | ",vetor[posicao]);
17
18        printf("\n\nExibido ao contrario, os elementos do vetor
sao: | | ");
19        for(posicao=9;posicao>=0;posicao--)
20            printf("%d | ",vetor[posicao]);
21
22        printf("\n\nOs elementos pares do vetor sao: | | ");
23        for(posicao=0; posicao<elementos; posicao++)
24            if((vetor[posicao] % 2) == 0)
25                printf("%d | ",vetor[posicao]);
26
27        printf("\n\nO maior elemento do vetor e: | ");
28        for(posicao=0; posicao<elementos; posicao++)
29            if(vetor[posicao] > maior)
30                maior = vetor[posicao];
31        printf("%d | ",maior);
32    }

```

Uma das maneiras mais fáceis de incluir elementos em um vetor é utilizando o laço for, o que é feito entre as linhas 10 e 12 do código acima.

Depois disso, o código do programa exibe os valores contidos no vetor, nas linhas 14 a 16, e exibe o vetor ao contrário entre as linhas 18 e 20.

Ainda utilizando o laço for, mas agora fazendo um teste com cada posição do vetor, são exibidos os elementos pares nas linhas 22 a 25.

Finalizando, nas linhas 27 a 31, é exibido o maior elemento do vetor, atribuindo cada valor de posição a uma variável, a qual recebe um valor maior cada vez que o teste feito é verdadeiro.

A execução do código acima exibirá uma tela semelhante à da Figura 4.

```
O vetor criado contem os seguintes elementos:  
! 0 ! 1 ! 2 ! 3 ! 4 ! 5 ! 6 ! 7 ! 8 ! 9 !  
Exibido ao contrario, os elementos do vetor sao:  
! 9 ! 8 ! 7 ! 6 ! 5 ! 4 ! 3 ! 2 ! 1 ! 0 !  
Os elementos pares do vetor sao:  
! 0 ! 2 ! 4 ! 6 ! 8 !  
O maior elemento do vetor e: ! 9 !  
Process exited after 3.66 seconds with return value 0  
Pressione qualquer tecla para continuar. . .
```

Figura 4. Execução do código.

Para que se possa compreender a aplicação de uma matriz, pode-se criar um arquivo em C com o nome *matriz.cpp*, cujo código será o seguinte:

1	#include <stdio.h>
2	#include <stdlib.h>
3	#include <string.h>
4	
5	#define linhas 4
6	#define colunas 8
7	
8	main() {
9	int posicao_linha = 0, posicao_coluna = 0, matriz[linhas][colunas], valor = 0, maior = 0;
10	
11	for(posicao_linha=0;posicao_linha<linhas;posicao_linha++) {
12	for(posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++) {
13	valor = posicao_linha + posicao_coluna; matriz[posicao_linha][posicao_coluna] = valor;
14	

```

15         }
16     }
17
18     printf("A matriz criada contem os seguintes elementos: \n\n"
19 " ");
20     for(posicao_linha=0; posicao_linha<linhas; posicao_linha++)
21 {
22     for(posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++)
23     {
24         printf("| %2d "
25         ",matriz[posicao_linha][posicao_coluna]);
26         }
27         printf("\n ");
28     }
29     printf("\n O maior elemento da matriz e: \n\n ");
30 }
```

```

27 {
28     for(posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++)
29     {
30         if(matriz[posicao_linha][posicao_coluna] >
31         maior)
32             maior =
33             matriz[posicao_linha][posicao_coluna];
34     }
35     printf("| %d | ",maior);
36 }
```

Como a matriz é formada de vários vetores, mas tem mais de uma dimensão, uma das melhores maneiras de inserir elementos em uma matriz é utilizar laços de for encadeados, nos quais um faz a varredura nas posições da linha e outro faz a varredura nas posições da coluna. Entre as linhas 11 e 16, o código faz a inserção de valores nas posições da matriz.

Logo a seguir, nas linhas 18 a 24, são utilizados laços de for encadeados, novamente, para fazer a exibição do valor contido em cada uma das posições da matriz.

Finalizando, entre as linhas 26 e 34 está o código utilizado para verificar qual é o maior elemento da matriz. Dessa forma, novamente são utilizados laços de for encadeados, mas, para isso, é feito um teste para cada um dos valores das posições para verificar se estes são maiores do que uma variável, a qual foi inicializada com 0, mas recebe um valor maior, caso seja encontrado (Figura 5).

```
A matriz criada contém os seguintes elementos:  
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |  
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |  
O maior elemento da matriz é:  
: 10 :  
Process exited after 3.974 seconds with return value 0  
Pressione qualquer tecla para continuar. . .
```

Figura 5. Exemplo de operações com matriz.

Ponteiros

Um ponteiro é um tipo de dado que se diferencia de uma variável, no sentido de que a variável faz uma referência direta a um valor. Já o ponteiro faz uma referência indireta a um valor. Quando se faz a modificação do valor de um ponteiro, está sendo modificado o valor da variável para a qual o ponteiro está apontando. Em outras palavras, o ponteiro é um tipo de dado que permite referenciar a posição de um objeto na memória, mas também o próprio objeto (DEITEL; DEITEL, 2011).

Quando se trabalha com ponteiros, o operador unário `*` é utilizado para acessar o conteúdo da variável para o qual o ponteiro está apontando, enquanto o operador unário `&` é utilizado para acessar o endereço da variável para o qual o ponteiro está apontando.

É importante lembrar que cada ponteiro deve ser declarado conforme o tipo de dado da variável para a qual ele vai apontar e que a declaração de um ponteiro é feita ao colocar um `*` na frente da variável:

```

1 #include <stdio.h>
2
3 main() {
4     int qtd_alunos = 1250;
5     int *pQtd_alunos = NULL;
6     pQtd_alunos = &qtd_alunos;
7
8     printf("A variavel idade vale %d.\n", qtd_alunos);
9     printf("O ponteiro acessando o valor da variavel vale
10    %d.\n", *pQtd_alunos);
11    printf("O ponteiro acessando o endereco da variavel vale
12    %d", pQtd_alunos);
13 }

```

No código acima, é feita a atribuição de um valor para uma variável. Depois, um ponteiro vai fazer a referência para essa variável, o que possibilita exibir o seu valor e o seu endereço.

A execução do código exibe uma tela semelhante à apresentada na Figura 6.

```

A variavel idade vale 1250.
O ponteiro acessando o valor da variavel vale 1250.
O ponteiro acessando o endereco da variavel vale 2358852
Process exited after 3.911 seconds with return value 0
Pressione qualquer tecla para continuar...

```

Figura 6. Exemplo de declaração de ponteiro.

É possível também fazer operações de soma e subtração com uma variável do tipo ponteiro e, nesse caso, será somada ou diminuída, no ponteiro, a quantidade de endereços de memória que é relativa ao tipo de dado que o ponteiro armazena (LAUREANO, 2008).

De maneira resumida, se um ponteiro que armazena um **int** ocupa 4 bytes, cada operação de soma feita no ponteiro vai acrescentar 4 posições ou unidades de memória. Já se o ponteiro armazenar um **char**, o qual ocupa 1 byte de memória, cada operação de soma feita no ponteiro vai acrescentar 1 posição ou unidade de memória. O exemplo abaixo ilustra essa explicação:

```
1 #include <stdio.h>
2
3 int main() {
4     int valor1;
5     int *pValor1;
6     char valor2;
7     char *pValor2;
8
9     pValor1 = &valor1;
10    pValor2 = &valor2;
11
12    printf ("Endereço de pValor1 = %d\n", pValor1);
13    pValor1++; //VAI SOMAR 4 BYTES NA MEMÓRIA PARA ESSA
14    VARIÁVEL
```

```
14    printf ("Endereço novo de pValor1 = %d\n", pValor1);
15
16    printf ("Endereço de pValor2 = %d\n", pValor2);
17    pValor2++; //VAI SOMAR 1 BYTE NA MEMÓRIA PARA ESSA
18    VARIÁVEL
19    printf ("Endereço novo de pValor2 = %d\n", pValor2);
20    //return 0;
```

A execução desse código vai retornar a uma tela semelhante à da Figura 7.

```
Enderoco de pValor1 = 2358844
Enderoco novo de pValor1 = 2358848
Enderoco de pValor2 = 2358843
Enderoco novo de pValor2 = 2358844

Process exited after 5.583 seconds with return value 0
Pressione qualquer tecla para continuar. . .
```

Figura 7. Exemplo de aritmética de ponteiro.



Referências

- DEITEL, P. J.; DEITEL, H. M. *Como programar em C*. 6. ed. São Paulo: Pearson, 2011.
- LAUREANO, M. *Estrutura de dados com algoritmos e C*. São Paulo: Brasport, 2008.
- TENENBAUM, A. A.; LANGSAM, Y.; AUGENSTEIN, M. J. *Estruturas de dados usando C*. São Paulo: Makron Books, 1995.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:



Dica do Professor

A área da computação trata diretamente com o armazenamento, a organização, a manipulação e a utilização de dados e informações, por isso é importante entender como esse processo ocorre.

O estudo da estrutura de dados em C envolve a identificação e o desenvolvimento de entidades e operações úteis para diversos tipos de aplicações, e a forma como fazer a sua abstração.

Na Dica do Professor a seguir, você vai aprender sobre as estruturas de dados.

Confira.



Aponte a câmera para o código e accese o link do conteúdo ou clique no código para accesar.

Exercícios

1) Qual a diferença entre dado e informação?

- A) Dados não têm significado de forma isolada e servem de base para a informação. A informação é fruto do processamento dos dados.
- B) Os dados são fruto do processamento das informações. A informação, de forma isolada, não tem significado algum.
- C) Dado e informação são sinônimos, ambos significando a maneira organizada e ordenada de armazenar o conhecimento.
- D) Dado e informação não têm vinculação alguma, sendo dado a maneira de armazenar o conhecimento, e informação o conhecimento em si.
- E) Dado é um tipo de dados na linguagem C, e informação é um sinônimo para estrutura de dados. *Os dados não têm significado relevante de maneira isolada e são considerados fundamentos da informação. A informação consiste na ordenação e na organização dos dados, para que seja possível compreender o seu significado, ou seja, ela é fruto do processamento dos dados.*

2) O que é um índice?

- A) É o nome de uma variável definida para o início de um laço de repetição FOR.
- B) É uma referência utilizada normalmente em estrutura de dados, que facilita o trabalho quando é feita uma consulta que envolve vários dados. *Um índice é um tipo de referência, normalmente utilizado quando se trabalha com estruturas de dados, que serve para otimizar localizações de registros em consultas com muitos dados.*
- C) É um tipo de estrutura de dados homogênea.
- D) É um tipo de estrutura de dados heterogênea.
- E) É a maneira de declarar uma estrutura em C: índice nome_estrutura {};

3) Qual a diferença entre estruturas de dados homogêneos e heterogêneos?

- A) Estruturas de dados heterogêneas armazenam o mesmo tipo de dados, e estruturas homogêneas armazenam tipos de dados diferentes.
- B) Estruturas de dados homogêneas são as que consistem em vetores, pois são unidimensionais, e as heterogêneas consistem em matrizes, pois são bidimensionais.

- C) Estruturas de dados homogêneas e heterogêneas são sinônimos e servem para varrer os dados de um vetor e de uma matriz em um laço FOR.
- D) Estruturas de dados homogêneas são vetores ordenados em ordem crescente, e heterogêneas são vetores ordenados em ordem decrescente.
- E) Estruturas de dados homogêneas armazenam o mesmo tipo de dados, e estruturas heterogêneas armazenam tipos de dados diferentes.

4) Quais são as estruturas de dados que representam o tipo de estrutura de dados homogêneos?

Quando uma estrutura de dados armazena elementos de tipos diferentes, é dito que é uma estrutura de dados heterogênea. Já uma estrutura de dados que armazene somente um tipo de dados é conhecida, então, como dados homogêneos. Os dados homogêneos são representados pelos vetores, ou estruturas de dados unidimensionais, e pelas matrizes, estruturas de dados bidimensionais.

- A) Pilhas e listas.
- B) Variáveis e índices.
- C) Filas e árvores. As estruturas de dados homogêneos são representadas pelos vetores, ou estruturas de dados unidimensionais, e pelas matrizes, estruturas de dados bidimensionais.
- D) Vetores e matrizes.
- E) Abstratas e concretas.

5) Por que um vetor é uma estrutura unidimensional, e uma matriz é uma estrutura bidimensional?

- A) Porque a matriz armazena dados de forma sequencial, e o vetor armazena dados dispostos em linhas e colunas.
- B) Porque o vetor armazena dados em linhas e colunas, e a matriz armazena dados em forma de índices.
- C) Porque o vetor armazena dados de forma sequencial, e a matriz armazena dados dispostos em linhas e colunas.
- D) Porque o vetor armazena dados em forma de estruturas, e a matriz armazena dados em forma de linhas sequenciais.
- E) Porque o vetor só possui uma posição [0], podendo armazenar um valor, e a matriz contém duas posições [0][0], podendo armazenar dois valores.

O vetor é uma estrutura unidimensional porque armazena dados em uma dimensão só, de forma sequencial. A matriz é considerada uma estrutura bidimensional porque é formada de vários vetores, ou seja, armazena dados em linhas e colunas.

Na prática

Quando uma estrutura de dados armazena elementos de tipos diferentes, é dito que se trata de uma estrutura de dados heterogênea. Já uma estrutura de dados que armazene somente um tipo de dados é conhecida, então, como de dados homogêneos.

Os dados homogêneos são representados pelos vetores, ou estruturas de dados unidimensionais, e pelas matrizes, estruturas de dados bidimensionais. Veja a seguir uma situação prática de dados heterogêneos e homogêneos.

O QUE É UMA ESTRUTURA DE DADOS?

Os vetores e as matrizes são estruturas de dados homogêneas, que possuem basicamente as seguintes características:

- São alocados de forma estática, ou seja, no momento da sua declaração, deve-se conhecer o tamanho ou a quantidade de posições que eles terão.
- São estruturas de dados homogêneas, formadas por elementos com o mesmo tipo de dados.
- Cada posição do vetor contém somente um valor.
- Os tamanhos dos valores de cada posição, por serem o mesmo tipo, são iguais.
- Fazem a alocação dos dados de forma sequencial.

Para inserir e localizar valores dentro de um vetor e de uma matriz, é necessário utilizar um índice para percorrer as suas posições. Existem outras maneiras de executar esse tipo de operação, mas, sem dúvida, uma das formas mais fáceis é utilizando laços de repetição do tipo for, que vão varrer a estrutura, inserindo ou lendo os dados, posição por posição, uma a uma.

É importante saber como executar essa simples atividade, pois ela serve de base para muitos tipos de aplicações desenvolvidas em sistemas computacionais.

Para executar a varredura das posições de um vetor, deve-se proceder da seguinte forma:

Para inserir os valores

```
//VARRE O VETOR, POSIÇÃO A POSIÇÃO
for(posicao=0;posicao<quantidade_de_elementos;posicao++){
    // INSERE UM VALOR NA POSIÇÃO ATUAL DO VETOR
    vetor[posicao] = posicao;
}
```

Para mostrar os valores do vetor:

```
printf("O vetor criado contém os seguintes elementos: \n\n| ");
//VARRE O VETOR, POSIÇÃO A POSIÇÃO
for(posicao=0; posicao< quantidade_de_elementos; posicao++){
    // MOSTRA CADA POSIÇÃO DO VETOR
    printf("%d | ",vetor[posicao]);
```

Para executar a varredura das posições de uma matriz, deve-se proceder da seguinte forma:

Para inserir valores na matriz:

```
//VARRE O VETOR LINHA DA MATRIZ, POSIÇÃO A POSIÇÃO
for(posicao_linha=0;posicao_linha<linhas;posicao_linha++){
//VARRE A COLUNA DA MATRIZ, POSIÇÃO A POSIÇÃO
for(posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++){
// COLOCA VALOR EM UMA VARIÁVEL E FAZ A POSIÇÃO DA MATRIZ RECEBER O VALOR
valor = posicao_linha + posicao_coluna;
matriz[posicao_linha][posicao_coluna] = valor;
}
}
```

Para mostrar os valores da matriz:

```
printf("A matriz criada contém os seguintes elementos: \n\n| ");
//VARRE O VETOR LINHA DA MATRIZ, POSIÇÃO A POSIÇÃO
for(posicao_linha=0; posicao_linha<linhas; posicao_linha++) {
//VARRE A COLUNA DA MATRIZ, POSIÇÃO A POSIÇÃO
for(posicao_coluna=0;posicao_coluna<colunas;posicao_coluna++){
    // MOSTRA CADA POSIÇÃO DA MATRIZ
    printf("| %2d ",matriz[posicao_linha][posicao_coluna]);
}
printf("|\n ");
}
```



Aponte a câmera para o código e acesse o link do conteúdo ou clique no código para acessar.

Saiba mais

Para ampliar o seu conhecimento a respeito desse assunto, veja abaixo as sugestões do professor:

Programar em C - Introdução Estruturas - Aula 39

Assista ao vídeo a seguir para saber mais sobre estruturas de dados em C.



Aponte a câmera para o código e accese o link do conteúdo ou clique no código para acessar.

Programar em C - Revisão Vetores/Matrizes - Aula 27

Assista ao vídeo a seguir para saber mais sobre vetores e matrizes em C.



Aponte a câmera para o código e accese o link do conteúdo ou clique no código para acessar.

Variáveis Compostas

Veja os tipos de variáveis compostas no link a seguir.



Aponte a câmera para o código e accese o link do conteúdo ou clique no código para acessar.