

Programação Imperativa

Prof. Dr. Alcides Calsavara

Escola Politécnica

PUCPR

Trabalhando com estruturas de dados: vetores

Motivação



Motivação

Considere uma empresa comercial que vende os seus produtos de segunda-feira a sábado. Para cada dia da semana, a empresa arrecada um valor (em reais) com as vendas, denominado de *faturamento*.

Escreva um programa que registre o faturamento de cada dia de uma semana e, então, determine o seguinte:

1. O faturamento total da semana
2. O faturamento diário médio
3. O dia da semana com o menor faturamento
4. O dia da semana com o maior faturamento

Motivação

Considere, para fins do programa, que o faturamento da empresa na semana em questão tenha ocorrido conforme mostra a seguinte tabela:

Dia da Semana	Faturamento (R\$)
segunda-feira	800.00
terça-feira	620.00
quarta-feira	900.00
quinta-feira	450.00
sexta-feira	800.00
sábado	780.00

O programa precisará definir uma variável (`float` ou `double`) para cada linha da tabela, isto é, para o faturamento de cada dia da semana.

Motivação

Neste programa, existem seis variáveis do tipo `float`: cada uma armazena o faturamento de um dia da semana.

Complete o programa conforme indicado pelos comentários.

```
#include <stdio.h>

int main()
{
    float faturamento__2 = 800.00; // segunda-feira
    float faturamento__3 = 620.00; // terça-feira
    float faturamento__4 = 900.00; // quarta-feira
    float faturamento__5 = 450.00; // quinta-feira
    float faturamento__6 = 800.00; // sexta-feira
    float faturamento__7 = 780.00; // sábado

    // Cálculo do faturamento semanal

    // Cálculo do faturamento diário médio

    // Identificação do dia da semana com o menor faturamento

    // Identificação do dia da semana com o maior faturamento

    return 0;
}
```

Motivação

```
// Cálculo do faturamento semanal
```

```
float faturamento_semanal = faturamento__2 + faturamento__3 + faturamento__4 +  
                             faturamento__5 + faturamento__6 + faturamento__7;
```

```
printf("Faturamento semanal: R$%.2f\n", faturamento_semanal);
```


Motivação

```
// Cálculo do faturamento semanal
```

```
float faturamento_semanal = faturamento__2 + faturamento__3 + faturamento__4 +  
                             faturamento__5 + faturamento__6 + faturamento__7;
```

```
printf("Faturamento semanal: R$%.2f\n", faturamento_semanal);
```

```
// Cálculo do faturamento diário médio
```

```
float faturamento_medio = faturamento_semanal / 6;
```

```
printf("Faturamento medio: R$%.2f\n", faturamento_medio);
```

Motivação

```
// Identificação do dia da semana com o menor faturamento

float menor_faturamento = faturamento__2;
int dia_menor = 2;

if (faturamento__3 < menor_faturamento) { menor_faturamento = faturamento__3; dia_menor = 3; }

if (faturamento__4 < menor_faturamento) { menor_faturamento = faturamento__4; dia_menor = 4; }

if (faturamento__5 < menor_faturamento) { menor_faturamento = faturamento__5; dia_menor = 5; }

if (faturamento__6 < menor_faturamento) { menor_faturamento = faturamento__6; dia_menor = 6; }

printf("Dia com o menor faturamento: %d\n", dia_menor);
printf("Menor faturamento: R$%.2f\n", menor_faturamento);
```


Motivação

```
// Identificação do dia da semana com o maior faturamento
```

```
float maior_faturamento = faturamento__2;  
int dia_maior = 2;
```

```
if (faturamento__3 > maior_faturamento) { maior_faturamento = faturamento__3; dia_maior = 3; }
```

```
if (faturamento__4 > maior_faturamento) { maior_faturamento = faturamento__4; dia_maior = 4; }
```

```
if (faturamento__5 > maior_faturamento) { maior_faturamento = faturamento__5; dia_maior = 5; }
```

```
if (faturamento__6 > maior_faturamento) { maior_faturamento = faturamento__6; dia_maior = 6; }
```

```
printf("Dia com o maior faturamento: %d\n", dia_maior);
```

```
printf("Maior faturamento: R$%.2f\n", maior_faturamento);
```

Motivação

Agora, adapte o programa construído para gerenciar o faturamento de uma semana da empresa para que passe a gerenciar o faturamento de um ano todo. Suponha que o ano tenha sempre 365 dias. O programa deverá determinar o seguinte:

1. O faturamento total do ano
2. O faturamento diário médio
3. O dia do ano com o menor faturamento
4. O dia do ano com o maior faturamento

Motivação

O programa precisará definir uma variável para cada dia do ano, ou seja, precisará de 365 variáveis.

O cálculo do faturamento anual terá uma expressão com a soma de 365 variáveis.

A determinação do dia do ano com o menor faturamento exigirá a programação de 364 comandos `if`.

A determinação do dia do ano com o maior faturamento exigirá a programação de 364 comandos `if`.



Usando vetores para representar uma sequência de dados

➡➡ array

Vetor: sequência de dados indexada

Índice	Dado
0	7.0
1	8.5
2	6.2
3	9.0
4	4.5
5	8.0

Vetor: sequência de dados indexada

0	408
1	175
2	941
3	388
4	539
5	650

0	7.0
1	8.5
2	6.2
3	9.0
4	4.5
5	8.0

0	true
1	true
2	false
3	true
4	false
5	true

Cada dado da sequência é dito ser um **elemento** do vetor.

Os elementos de um vetor são de um único tipo.

A **capacidade** de um vetor é a quantidade máxima de elementos que cabem no vetor. É um valor fixo.

O **índice** é um valor inteiro não negativo, sendo *zero* o índice do primeiro elemento.

Declaração de um vetor na linguagem C

Forma:

```
tipo identificador [capacidade];
```

tipo: tipo dos elementos (int, float, char, bool, etc)

identificador: variável que representa a sequência de elementos

capacidade: número máximo de elementos na sequência

Exemplos:

```
float faturamento_semana[6];
```

```
float faturamento_ano[365];
```

Exemplos de vetores

0	408
1	175
2	941
3	388
4	539
5	650

```
int matricula[6];
```

0	7.0
1	8.5
2	6.2
3	9.0
4	4.5
5	8.0

```
float nota[6];
```

0	true
1	true
2	false
3	true
4	false
5	true

```
bool aprovado[6];
```

Declaração e inicialização de um vetor

Modo 1:

```
float nota[6];
```

```
nota[0] = 7.0;  
nota[1] = 8.5;  
nota[2] = 6.2;  
nota[3] = 9.0;  
nota[4] = 4.5;  
nota[5] = 8.0;
```

Modo 2:

```
float nota[6] = { 7.0, 8.5, 6.2, 9.0, 4.5, 8.0 };
```

Modo 2:

O vetor é declarado e todos os elementos são imediatamente definidos por meio de uma sequência de valores.

Modo 1:

- a) O vetor é declarado, deixando os valores dos elementos indefinidos.
- b) O valor de cada elemento é definido posteriormente.

Declaração e inicialização de um vetor

Modo 3:

```
float nota[6] = { 7.0, 8.5, 6.2, 9.0 };
```

Modo 3:

O vetor é declarado e somente parte dos elementos são imediatamente definidos por meio de uma sequência de valores.

Modo 4:

```
float nota[ ] = { 7.0, 8.5, 6.2, 9.0 };
```

Modo 4:

O vetor é declarado e todos os elementos são imediatamente definidos por meio de uma sequência de valores, sendo a sua capacidade determinada pelo número de valores na sequência.

Impressão de um vetor

```
float nota[6] = { 7.0, 8.5, 6.2, 9.0, 4.5, 8.0 };
```

Modo 1:

```
printf("%.1f ", nota[0]);  
printf("%.1f ", nota[1]);  
printf("%.1f ", nota[2]);  
printf("%.1f ", nota[3]);  
printf("%.1f ", nota[4]);  
printf("%.1f ", nota[5]);
```

Modo 2:

```
for (int i = 0; i < 6; i++)  
    printf("%.1f ", nota[i]);
```

Teste!

Leitura dos elementos de um vetor

```
float nota[6];
```

Leitura de um elemento específico:

```
puts("Digite a primeira nota:");  
scanf("%f", &nota[0]);
```

Leitura de todos os elementos:

```
for (int i = 0; i < 6; i++)  
{  
    printf("Digite o valor do elemento %d: ", i);  
    scanf("%f", &nota[i]);  
}
```

Teste!

Cálculos com os elementos de um vetor

```
float nota[6] = { 7.0, 8.5, 6.2, 9.0, 4.5, 8.0 };
```

```
float soma = 0;

for (int i = 0; i < 6; i++)
    soma = soma + nota[i];

printf("soma: %.2f\n", soma);

float media = soma / 6;

printf("media: %.2f\n", media);
```

Teste!

Identificação do menor elemento

```
float nota[6] = { 7.0, 8.5, 6.2, 9.0, 4.5, 8.0 };
```

```
float menor_nota = nota[0];  
int indice_menor_nota = 0;  
  
for (int i = 1; i < 6; i++)  
    if (nota[i] < menor_nota)  
    {  
        menor_nota = nota[i];  
        indice_menor_nota = i;  
    }  
  
printf("Menor nota: %.2f\n", menor_nota);  
printf("Indice da menor nota: %d\n", indice_menor_nota);
```

Teste!

Determinação de tamanho e capacidade

```
float nota[ ] = { 7.0, 8.5, 6.2, 9.0, 4.5, 8.0 };
```

```
int tamanho_vetor = sizeof(nota);  
printf("Tamanho do vetor: %zu bytes\n", tamanho_vetor);  
  
int tamanho_elemento = sizeof(nota[0]);  
printf("Tamanho de um elemento: %zu bytes\n", tamanho_elemento);  
  
int capacidade_vetor = tamanho_vetor / tamanho_elemento;  
printf("Logo, o vetor possui %d elementos.\n", capacidade_vetor);
```

Teste!

Acesso além do limite de um vetor

```
#include <stdio.h>

int main()
{
    int primo[4] = { 2, 3, 5, 7 };

    primo[4] = 11; ← consequências imprevisíveis

    for (int i = 0; i < 5; i++)
        printf("primo[%d]: %d\n", i, primo[i]);

    return 0;
}
```

Exemplo: vetor de elementos booleanos

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

```
bool b[ ] = {true, false, false, true, true, false};
```

```
int num_elementos = sizeof(b) / sizeof(bool); ← sizeof(b) / sizeof( b[0] )
printf("Numero de elementos: %d\n", num_elementos);
```

```
for (int i = 0; i < num_elementos; i++)
{
    if (b[i])
        puts("verdadeiro");
    else
        puts("falso");
}
```

Exemplo: vetor de caracteres

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

Biblioteca que define a função `setbuf` e outras.

```
printf("Digite o comprimento da string: ");
int comprimento;
scanf("%d", &comprimento);

char cadeia[comprimento];

printf("Digite a string: ");

setbuf(stdin, NULL); // limpa o buffer do teclado

for (int i = 0; i < comprimento; i++)
{
    cadeia[i] = getchar();
    putchar(cadeia[i]);
}
```

Teste este código como está e, depois, teste-o eliminando a chamada da função `setbuf`.

Teste a seguinte entrada:

Digite o comprimento da string:
SP SP 5 SP SP ENTER
Digite a string:
abcdefg ENTER

array_char.c

Vetores e endereços de memória



O endereço de um vetor

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

```
int main()  
{
```

```
    int quantidade;  
    float preco;  
    int estoque[10];  
    char palavra[10];
```

```
    printf("%lu\n", palavra);  
    printf("%lu\n", estoque);  
    printf("%lu\n", &preco);  
    printf("%lu\n", &quantidade);
```

```
    return 0;
```

```
}
```

Não se usa o & para o endereço de um vetor; o identificador do vetor corresponde ao seu endereço.

6 422 246

10 bytes

palavra

6 422 256

40 bytes

estoque

6 422 296

4 bytes

preco

6 422 300

4 bytes

quantidade

O endereço de cada elemento

```
for (int i = 0; i < 10; i++)  
    printf("endereço de palavra[%d]: %lu\n", i, &palavra[i]);
```

6 422 246		palavra[0]
6 422 247		palavra[1]
6 422 248		palavra[2]
6 422 249		palavra[3]
6 422 250		palavra[4]
6 422 251		palavra[5]
6 422 252		palavra[6]
6 422 253		palavra[7]
6 422 254		palavra[8]
6 422 255		palavra[9]

Exercício 7.1

Considere uma empresa comercial que vende os seus produtos de segunda-feira a sábado. Para cada dia da semana, a empresa arrecada um valor (em reais) com as vendas, denominado de *faturamento*.

Escreva um programa que registre o faturamento de cada dia de uma semana e, então, determine o seguinte:

1. O faturamento total da semana
2. O faturamento diário médio
3. O dia da semana com o menor faturamento
4. O dia da semana com o maior faturamento

O programa deve utilizar um vetor para armazenar o faturamento da semana: cada elemento do vetor corresponde ao faturamento de um dia.

Exercícios

7.2 Escreva um programa em C que leia do teclado dois vetores (A e B) com cinco elementos inteiros cada um, gere um terceiro vetor (S) com a soma dos correspondentes elementos dos dois primeiros ($S[i] = A[i] + B[i]$) e imprima os três vetores.

7.3 Escreva um programa em C que leia do teclado um vetor de oito elementos reais, imprima esse vetor e, depois, imprima a soma dos elementos equidistantes, isto é, imprime a soma do primeiro com último, depois a soma do segundo com o penúltimo, e assim por diante.

7.4 Escreva um programa em C que leia do teclado um vetor de caracteres, gere um novo vetor de caracteres na ordem inversa do primeiro e imprima os dois vetores. A quantidade de caracteres (isto é, a capacidade do vetor) deve ser determinada pelo usuário (valor fornecido via teclado).

7.5 Escreva um programa em C que leia do teclado uma sequência de caracteres de qualquer tamanho e identifique se é ou não um **palíndromo**. Antes de fornecer a sequência de caracteres, o usuário deve informar o tamanho da sequência.

Exercícios

7.6 Escreva um programa em C que leia do teclado uma sequência qualquer de valores inteiros e armazene-a em um vetor *v*. Em seguida, leia do teclado um valor inteiro *x* e procure esse valor no vetor *v*. O programa deve informar se *x* foi ou não encontrado em *v*. Se encontrado, deve informar o índice do vetor onde *x* se encontra.

7.7 Escreva um programa em C que leia do teclado uma sequência qualquer de valores inteiros e armazene-a em um vetor *v*. Em seguida, gere um novo vetor de inteiros com a mesma capacidade e os mesmos elementos de *v*, porém em ordem crescente.

7.8 Escreva um programa em C que leia do teclado uma sequência qualquer de valores inteiros distintos entre si e armazene-a em um vetor. Para cada valor da sequência fornecido pelo usuário, o programa deve verificar se, de fato, é um valor inédito antes de inseri-lo no vetor. Caso não seja, o programa deve rejeitar o valor fornecido.

7.9 Escreva um programa em C que calcule a média e o desvio padrão das notas de uma turma de, no máximo, 50 estudantes. Para cada estudante, há uma nota entre 0 e 10, com apenas uma casa decimal.