Vetores

Aula 12

Fábio Henrique Viduani Martinez

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação I, 2012

Conteúdo da aula

Motivação

- tipos primitivos de dados se caracterizam pelo fato que seus valores não podem ser decompostos
- ou seja, um valor armazenado em uma variável de um tipo primitivo é único e não faz parte de uma composição de valores organizada de alguma maneira
- tipos primitivos de dados também são chamados de básicos ou elementares
- se os valores de um tipo de dados podem ser decompostos ou subdivididos em valores mais simples, então o tipo de dados é chamado de complexo, composto ou estruturado
- a organização desses valores e as relações estabelecidas entre eles determinam o que conhecemos como estrutura de dados

Motivação

- tipos primitivos de dados se caracterizam pelo fato que seus valores não podem ser decompostos
- ou seja, um valor armazenado em uma variável de um tipo primitivo é único e não faz parte de uma composição de valores organizada de alguma maneira
- tipos primitivos de dados também são chamados de básicos ou elementares
- se os valores de um tipo de dados podem ser decompostos ou subdivididos em valores mais simples, então o tipo de dados é chamado de complexo, composto ou estruturado
- a organização desses valores e as relações estabelecidas entre eles determinam o que conhecemos como estrutura de dados

Motivação

- tipos primitivos de dados se caracterizam pelo fato que seus valores não podem ser decompostos
- ou seja, um valor armazenado em uma variável de um tipo primitivo é único e não faz parte de uma composição de valores organizada de alguma maneira
- tipos primitivos de dados também são chamados de básicos ou elementares
- se os valores de um tipo de dados podem ser decompostos ou subdivididos em valores mais simples, então o tipo de dados é chamado de complexo, composto ou estruturado
- a organização desses valores e as relações estabelecidas entre eles determinam o que conhecemos como estrutura de dados

Motivação

- tipos primitivos de dados se caracterizam pelo fato que seus valores não podem ser decompostos
- ou seja, um valor armazenado em uma variável de um tipo primitivo é único e não faz parte de uma composição de valores organizada de alguma maneira
- tipos primitivos de dados também são chamados de básicos ou elementares
- se os valores de um tipo de dados podem ser decompostos ou subdivididos em valores mais simples, então o tipo de dados é chamado de complexo, composto ou estruturado
- a organização desses valores e as relações estabelecidas entre eles determinam o que conhecemos como estrutura de dados

Motivação

- tipos primitivos de dados se caracterizam pelo fato que seus valores não podem ser decompostos
- ou seja, um valor armazenado em uma variável de um tipo primitivo é único e não faz parte de uma composição de valores organizada de alguma maneira
- tipos primitivos de dados também são chamados de básicos ou elementares
- se os valores de um tipo de dados podem ser decompostos ou subdivididos em valores mais simples, então o tipo de dados é chamado de complexo, composto ou estruturado
- a organização desses valores e as relações estabelecidas entre eles determinam o que conhecemos como estrutura de dados

Motivação

Problema: dadas cinco notas de uma prova dos(as) estudantes de uma disciplina, calcular a média das notas da prova e a quantidade de estudantes que obtiveram nota maior que a média e a quantidade de estudantes que obtiveram nota menor que a média

```
#include <stdio.h>

int main(void)
{
   int menor, maior;
   float nota1, nota2, nota3, nota4, nota5, media;

   printf("Informe as notas dos alunos: ");
   scanf("%f%f%f%f%f", &nota1, &nota2, &nota3, &nota4, &nota5);
   media = (nota1 + nota2 + nota3 + nota4 + nota5) / 5;
   printf("Media das provas: %f\n", media);
```

Motivação

Problema: dadas cinco notas de uma prova dos(as) estudantes de uma disciplina, calcular a média das notas da prova e a quantidade de estudantes que obtiveram nota maior que a média e a quantidade de estudantes que obtiveram nota menor que a média

```
#include <stdio.h>

int main(void)
{
   int menor, maior;
   float nota1, nota2, nota3, nota4, nota5, media;

   printf("Informe as notas dos alunos: ");
   scanf("%f%f%f%f%f", &nota1, &nota2, &nota3, &nota4, &nota5);
   media = (nota1 + nota2 + nota3 + nota4 + nota5) / 5;
   printf("Media das provas: %f\n", media);
```

Motivação

```
menor = 0;
if (nota1 < media)
   menor = menor + 1;
if (nota2 < media)
   menor = menor + 1;
if (nota3 < media)
   menor = menor + 1;
if (nota4 < media)
   menor = menor + 1;
if (nota5 < media)
   menor = menor + 1;
```

Motivação

```
    maior = 0;
    if (nota1 > media)
       maior = maior + 1;
    if (nota2 > media)
       maior = maior + 1;
    if (nota3 > media)
       maior = maior + 1;
    if (nota4 > media)
       maior = maior + 1;
    if (nota5 > media)
       maior = maior + 1;
    printf("Quantidade com nota inferior à média: %d\n", menor);
    printf("Quantidade com nota superior à média: %d\n", maior);

    return 0;
}
```

Motivação

- o que aconteceria se a sala de aula tivesse mais estudantes, como por exemplo 100? ou 1000 estudantes?
- as estruturas sequencial e condicional não seriam apropriadas para resolver esse problema, já que o programa teria centenas ou milhares de linhas repetitivas, incorrendo inclusive na possibilidade de propagação de erros e na dificuldade de encontrá-los

Motivação

- o que aconteceria se a sala de aula tivesse mais estudantes, como por exemplo 100? ou 1000 estudantes?
- as estruturas sequencial e condicional não seriam apropriadas para resolver esse problema, já que o programa teria centenas ou milhares de linhas repetitivas, incorrendo inclusive na possibilidade de propagação de erros e na dificuldade de encontrá-los

Motivação

- o que aconteceria se a sala de aula tivesse mais estudantes, como por exemplo 100? ou 1000 estudantes?
- as estruturas sequencial e condicional não seriam apropriadas para resolver esse problema, já que o programa teria centenas ou milhares de linhas repetitivas, incorrendo inclusive na possibilidade de propagação de erros e na dificuldade de encontrá-los

Definição

- uma variável composta homogênea unidimensional, ou simplesmente um vetor, é uma estrutura de armazenamento de dados que se dispõe de forma linear na memória e é usada para armazenar valores de um mesmo tipo
- um vetor é então uma lista de células na memória de tamanho fixo cujos conteúdos são do mesmo tipo
- cada uma dessas células armazena um, e apenas um, valor
- cada célula do vetor tem um endereço ou índice através do qual podemos referenciá-la
- a forma geral de declaração de um vetor na linguagem C é:

```
tipo identificador[dimensão];
```

Definição

- uma variável composta homogênea unidimensional, ou simplesmente um vetor, é uma estrutura de armazenamento de dados que se dispõe de forma linear na memória e é usada para armazenar valores de um mesmo tipo
- um vetor é então uma lista de células na memória de tamanho fixo cujos conteúdos são do mesmo tipo
- cada uma dessas células armazena um, e apenas um, valor
- cada célula do vetor tem um endereço ou índice através do qual podemos referenciá-la
- a forma geral de declaração de um vetor na linguagem C é:

```
tipo identificador[dimensão];
```

Definição

- uma variável composta homogênea unidimensional, ou simplesmente um vetor, é uma estrutura de armazenamento de dados que se dispõe de forma linear na memória e é usada para armazenar valores de um mesmo tipo
- um vetor é então uma lista de células na memória de tamanho fixo cujos conteúdos são do mesmo tipo
- cada uma dessas células armazena um, e apenas um, valor
- cada célula do vetor tem um endereço ou índice através do qual podemos referenciá-la
- a forma geral de declaração de um vetor na linguagem C é:

```
tipo identificador[dimensão];
```

Definição

- uma variável composta homogênea unidimensional, ou simplesmente um vetor, é uma estrutura de armazenamento de dados que se dispõe de forma linear na memória e é usada para armazenar valores de um mesmo tipo
- um vetor é então uma lista de células na memória de tamanho fixo cujos conteúdos são do mesmo tipo
- cada uma dessas células armazena um, e apenas um, valor
- cada célula do vetor tem um endereço ou índice através do qual podemos referenciá-la
- a forma geral de declaração de um vetor na linguagem C é:

```
tipo identificador[dimensão];
```

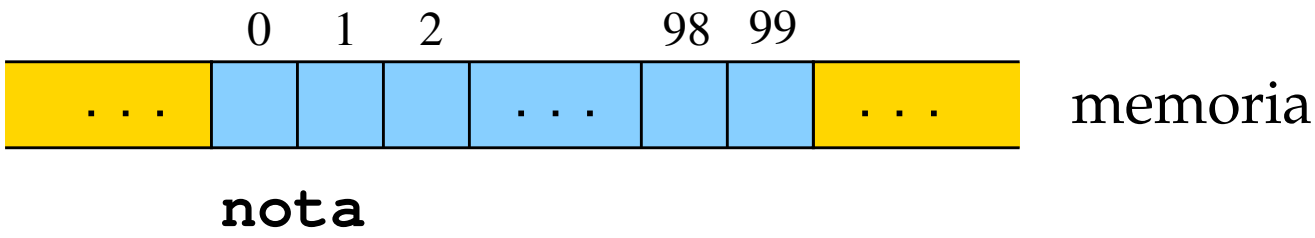
Definição

- uma variável composta homogênea unidimensional, ou simplesmente um vetor, é uma estrutura de armazenamento de dados que se dispõe de forma linear na memória e é usada para armazenar valores de um mesmo tipo
- um vetor é então uma lista de células na memória de tamanho fixo cujos conteúdos são do mesmo tipo
- cada uma dessas células armazena um, e apenas um, valor
- cada célula do vetor tem um endereço ou índice através do qual podemos referenciá-la
- a forma geral de declaração de um vetor na linguagem C é:

```
tipo identificador[dimensão];
```

Definição

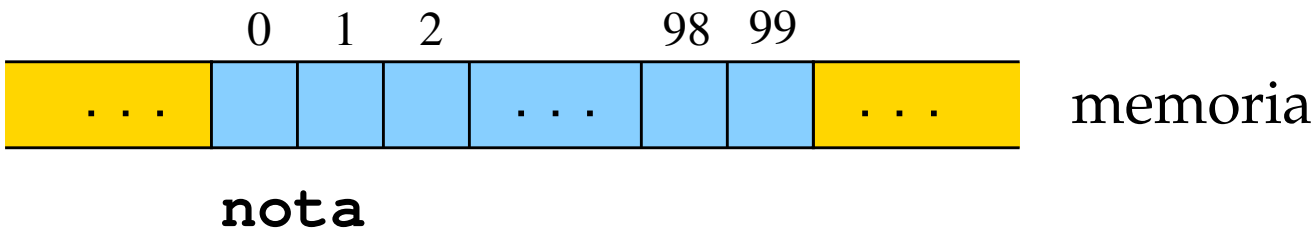
```
float nota[100];
```




```
nota[0]
nota[35]
nota[i]
nota[2 * i + j]
```

Definição

```
float nota[100];
```




```
nota[0]
nota[35]
nota[i]
nota[2 * i + j]
```

Definição

- o compilador da linguagem C não verifica de antemão se os limites dos índices de um vetor estão corretos

```
int A[10], i;
for (i = 1; i <= 10; i++)
   A[i] = 0;
```

Definição

- o compilador da linguagem C não verifica de antemão se os limites dos índices de um vetor estão corretos

```
int A[10], i;
for (i = 1; i <= 10; i++)
   A[i] = 0;
```

Declaração com inicialização

- podemos atribuir valores iniciais a quaisquer variáveis de qualquer tipo primitivo no momento de suas respectivas declarações

```
char c = 'a';
int num, soma = 0;
float produto = 1.0, resultado;
```

- uma forma comum de se fazer a declaração e inicialização de um vetor é através de uma lista de expressões constantes envolvidas por chaves e separadas por vírgulas:

```
int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Declaração com inicialização

- podemos atribuir valores iniciais a quaisquer variáveis de qualquer tipo primitivo no momento de suas respectivas declarações

```
char c = 'a';
int num, soma = 0;
float produto = 1.0, resultado;
```

- uma forma comum de se fazer a declaração e inicialização de um vetor é através de uma lista de expressões constantes envolvidas por chaves e separadas por vírgulas:

```
int A[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Declaração com inicialização

- outros exemplos:

```
int A[10] = {1, 2, 3, 4};
```

```
int A[10] = {1, 2, 3, 4, 0, 0, 0, 0, 0, 0};
```

```
int A[10] = {0};
```

```
int A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Declaração com inicialização

- outros exemplos:

```
int A[10] = {1, 2, 3, 4};
```

```
int A[10] = {1, 2, 3, 4, 0, 0, 0, 0, 0, 0};
```

```
int A[10] = {0};
```

```
int A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Declaração com inicialização

- outros exemplos:

```
int A[10] = {1, 2, 3, 4};
```

```
int A[10] = {1, 2, 3, 4, 0, 0, 0, 0, 0, 0};
```

```
int A[10] = {0};
```

```
int A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Declaração com inicialização

- outros exemplos:

```
int A[10] = {1, 2, 3, 4};
```

```
int A[10] = {1, 2, 3, 4, 0, 0, 0, 0, 0, 0};
```

```
int A[10] = {0};
```

```
int A[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

Exemplo com vetores

```
#include <stdio.h>
int main(void)
{
   int i, menor, maior;
   float nota[5], soma, media;

   for (i = 0; i < 5; i++)
      scanf("%f", &nota[i]);
   soma = 0.0;
   for (i = 0; i < 5; i++)
      soma = soma + nota[i];
   media = soma / 5;
   menor = 0;
   maior = 0;
   for (i = 0; i < 5; i++) {
      if (nota[i] < media)
         menor++;
      if (nota[i] > media)
         maior++;
   }
   printf("\n%2.2f %d %d\n", media, menor, maior);
   return 0;
}
```

Macros para constantes

- quando um programa contém constantes, uma boa idéia é dar nomes a essas constantes
- podemos atribuir um nome ou identificador a uma constante usando a definição de uma macro
- uma macro é definida através da diretiva de pré-processador `#define` e tem o seguinte formato geral:

```
#define identificador constante
```

- exemplos:

```
#define CARAC      'a'
#define NUMERADOR  4
#define MIN        -10000
#define TAXA       0.01567
```

Macros para constantes

- quando um programa contém constantes, uma boa idéia é dar nomes a essas constantes
- podemos atribuir um nome ou identificador a uma constante usando a definição de uma macro
- uma macro é definida através da diretiva de pré-processador `#define` e tem o seguinte formato geral:

```
#define identificador constante
```

- exemplos:

```
#define CARAC      'a'
#define NUMERADOR  4
#define MIN        -10000
#define TAXA       0.01567
```

Macros para constantes

- quando um programa contém constantes, uma boa idéia é dar nomes a essas constantes
- podemos atribuir um nome ou identificador a uma constante usando a definição de uma macro
- uma macro é definida através da diretiva de pré-processador `#define` e tem o seguinte formato geral:

```
#define identificador constante
```

- exemplos:

```
#define CARAC      'a'
#define NUMERADOR  4
#define MIN        -10000
#define TAXA       0.01567
```

Macros para constantes

- quando um programa contém constantes, uma boa idéia é dar nomes a essas constantes
- podemos atribuir um nome ou identificador a uma constante usando a definição de uma macro
- uma macro é definida através da diretiva de pré-processador `#define` e tem o seguinte formato geral:

```
#define identificador constante
```

- exemplos:

```
#define CARAC      'a'
#define NUMERADOR  4
#define MIN        -10000
#define TAXA       0.01567
```

Macros para constantes

- quando um programa é compilado, o pré-processador troca cada macro definida no código pelo valor que ela representa; depois disso, um segundo passo de compilação é executado
- o uso de macros com vetores é bastante útil porque um vetor faz alocação estática da memória, o que significa que em sua declaração ocorre uma reserva prévia de um número fixo de compartimentos de memória
- por ser uma alocação estática, não há possibilidade de aumento ou diminuição dessa quantidade após a execução da linha de código contendo a declaração do vetor

Macros para constantes

- quando um programa é compilado, o pré-processador troca cada macro definida no código pelo valor que ela representa; depois disso, um segundo passo de compilação é executado
- o uso de macros com vetores é bastante útil porque um vetor faz alocação estática da memória, o que significa que em sua declaração ocorre uma reserva prévia de um número fixo de compartimentos de memória
- por ser uma alocação estática, não há possibilidade de aumento ou diminuição dessa quantidade após a execução da linha de código contendo a declaração do vetor

Macros para constantes

- quando um programa é compilado, o pré-processador troca cada macro definida no código pelo valor que ela representa; depois disso, um segundo passo de compilação é executado
- o uso de macros com vetores é bastante útil porque um vetor faz alocação estática da memória, o que significa que em sua declaração ocorre uma reserva prévia de um número fixo de compartimentos de memória
- por ser uma alocação estática, não há possibilidade de aumento ou diminuição dessa quantidade após a execução da linha de código contendo a declaração do vetor

Macros para constantes

```
#include <stdio.h>
#define MAX 5
int main(void)
{
   int i, menor, maior;
   float nota[MAX], soma, media;
   for (i = 0; i < MAX; i++)
      scanf("%f", &nota[i]);
   soma = 0.0;
   for (i = 0; i < MAX; i++)
      soma = soma + nota[i];
   media = soma / MAX;
   menor = 0;
   maior = 0;
   for (i = 0; i < MAX; i++) {
      if (nota[i] < media)
         menor++;
      if (nota[i] > media)
         maior++;
   }
   printf("\n%2.2f %d %d\n", media, menor, maior);
   return 0;
}
```

Exercícios

1. Dada uma sequência de n números inteiros, com $1 \leqslant n \leqslant 100$, imprimi-la em ordem inversa à de leitura.

```c
#include <stdio.h>

#define MAX 100

int main(void)
{
   int i, n, A[MAX];

   scanf("%d", &n);
   for (i = 0; i < n; i++)
      scanf("%d", &A[i]);
   for (i = n-1; i >= 0; i--)
      printf("%d ", A[i]);
   printf("\n");

   return 0;
}
```

Exercícios

1. Dada uma sequência de n números inteiros, com $1 \leqslant n \leqslant 100$, imprimi-la em ordem inversa à de leitura.

```
#include <stdio.h>

#define MAX 100

int main(void)
{
   int i, n, A[MAX];

   scanf("%d", &n);
   for (i = 0; i < n; i++)
      scanf("%d", &A[i]);
   for (i = n-1; i >= 0; i--)
      printf("%d ", A[i]);
   printf("\n");

   return 0;
}
```

Exercícios

2. Uma prova consta de 20 questões, cada uma com cinco alternativas identificadas pelas letras A, B, C, D e E. Dado o cartão gabarito da prova e o cartão de respostas de n estudantes, com $1 \leqslant n \leqslant 100$, computar o número de acertos de cada um dos estudantes.
3. Tentando descobrir se um dado era viciado, um dono de cassino o lançou n vezes. Dados os n resultados dos lançamentos, determinar o número de ocorrências de cada face.
   Exemplo: se $n = 8$ e os resultados dos lançamentos são $6, 5, 3, 1, 6, 1, 2, 4$, então a saída deve ser $2, 1, 1, 1, 1, 2$.

Exercícios

2. Uma prova consta de 20 questões, cada uma com cinco alternativas identificadas pelas letras A, B, C, D e E. Dado o cartão gabarito da prova e o cartão de respostas de n estudantes, com $1 \leqslant n \leqslant 100$, computar o número de acertos de cada um dos estudantes.
3. Tentando descobrir se um dado era viciado, um dono de cassino o lançou n vezes. Dados os n resultados dos lançamentos, determinar o número de ocorrências de cada face.
   Exemplo: se $n = 8$ e os resultados dos lançamentos são $6, 5, 3, 1, 6, 1, 2, 4$, então a saída deve ser $2, 1, 1, 1, 1, 2$.

Exercícios

4. Um jogador viciado de cassino deseja fazer um levantamento estatístico simples sobre uma roleta. Para isso, ele fez n lançamentos nesta roleta. Sabendo que uma roleta contém 37 números (de 0 a 36), calcular a frequência de cada número desta roleta nos n lançamentos realizados.
   Exemplo: se $n = 10$ e os resultados dos lançamentos são 32, 12, 6, 0, 9, 12, 11, 7, 28, 21, então a saída deve ser 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0.
5. Dados dois vetores x e y, ambos com n elementos, $1 \leqslant n \leqslant 100$, determinar o produto escalar desses vetores.
   O **produto escalar** entre $\vec{u}$ e $\vec{v}$ é escrito como sendo:

$$\vec{u} \cdot \vec{v} = \sum_{i=0}^{n-1} u_i v_i = u_0 v_0 + u_1 v_1 + \cdots + u_{n-1} v_{n-1} \cdot$$

Exercícios

4. Um jogador viciado de cassino deseja fazer um levantamento estatístico simples sobre uma roleta. Para isso, ele fez n lançamentos nesta roleta. Sabendo que uma roleta contém 37 números (de 0 a 36), calcular a frequência de cada número desta roleta nos n lançamentos realizados.
   Exemplo: se $n = 10$ e os resultados dos lançamentos são 32, 12, 6, 0, 9, 12, 11, 7, 28, 21, então a saída deve ser 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0.
5. Dados dois vetores x e y, ambos com n elementos, $1 \leqslant n \leqslant 100$, determinar o produto escalar desses vetores.
   O **produto escalar** entre $\vec{u}$ e $\vec{v}$ é escrito como sendo:

$$\vec{u} \cdot \vec{v} = \sum_{i=0}^{n-1} u_i v_i = u_0 v_0 + u_1 v_1 + \cdots + u_{n-1} v_{n-1} \, .$$

Exercícios

6. Calcule o valor do polinômio $p(x) = a_0 + a_1 x + \ldots + a_n x^n$ em k pontos distintos. São dados os valores de n (grau do polinômio), com $1 \leqslant n \leqslant 100$, de $a_0, a_1, \ldots a_n$ (coeficientes reais do polinômio), de k e dos pontos $x_1, x_2, \ldots, x_k$.
7. Dado o polinômio $p(x) = a_0 + a_1 x + \ldots a_n x^n$, isto é, os valores de n e de $a_0, a_1, \ldots, a_n$, com $1 \leqslant n \leqslant 100$ determine os coeficientes reais da primeira derivada de $p(x)$.
8. Dados dois polinômios reais

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n \quad \text{e} \quad q(x) = b_0 + b_1 x + \ldots + b_m x^m$$

determinar o produto desses dois polinômios. Suponha que $1 \leqslant m, n \leqslant 100$.

Exercícios

6. Calcule o valor do polinômio $p(x) = a_0 + a_1 x + \ldots + a_n x^n$ em k pontos distintos. São dados os valores de n (grau do polinômio), com $1 \leqslant n \leqslant 100$, de $a_0, a_1, \ldots a_n$ (coeficientes reais do polinômio), de k e dos pontos $x_1, x_2, \ldots, x_k$.
7. Dado o polinômio $p(x) = a_0 + a_1 x + \ldots a_n x^n$, isto é, os valores de n e de $a_0, a_1, \ldots, a_n$, com $1 \leqslant n \leqslant 100$ determine os coeficientes reais da primeira derivada de $p(x)$.
8. Dados dois polinômios reais

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n \quad \text{e} \quad q(x) = b_0 + b_1 x + \ldots + b_m x^m$$

determinar o produto desses dois polinômios. Suponha que $1 \leqslant m, n \leqslant 100$.

Exercícios

6. Calcule o valor do polinômio $p(x) = a_0 + a_1 x + \ldots + a_n x^n$ em k pontos distintos. São dados os valores de n (grau do polinômio), com $1 \leqslant n \leqslant 100$, de $a_0, a_1, \ldots a_n$ (coeficientes reais do polinômio), de k e dos pontos $x_1, x_2, \ldots, x_k$.
7. Dado o polinômio $p(x) = a_0 + a_1 x + \ldots a_n x^n$, isto é, os valores de n e de $a_0, a_1, \ldots, a_n$, com $1 \leqslant n \leqslant 100$ determine os coeficientes reais da primeira derivada de $p(x)$.
8. Dados dois polinômios reais

$$p(x) = a_0 + a_1 x + \ldots + a_n x^n \quad \text{e} \quad q(x) = b_0 + b_1 x + \ldots + b_m x^m$$

determinar o produto desses dois polinômios. Suponha que $1 \leqslant m, n \leqslant 100$.

Exercícios

9. Dadas duas sequências com n números inteiros entre 0 e 9, interpretadas como dois números inteiros de n algarismos, $1 \leqslant n \leqslant 100$, calcular a sequência de números que representa a soma dos dois inteiros.
   Exemplo:
   $n = 8$,

1ª sequência		8	2	4	3	4	2	5	1
2ª sequência	+	3	3	7	5	2	3	3	7
		1	1	6	1	8	6	5	8	8

10. Dados dois números naturais m e n, com $1 \leqslant m, n \leqslant 100$, e duas sequências ordenadas com m e n números inteiros, obter uma única sequência ordenada contendo todos os elementos das sequências originais sem repetição.

Exercícios

9. Dadas duas sequências com n números inteiros entre 0 e 9, interpretadas como dois números inteiros de n algarismos, $1 \leqslant n \leqslant 100$, calcular a sequência de números que representa a soma dos dois inteiros.
   Exemplo:
   $n = 8$,

1ª sequência		8	2	4	3	4	2	5	1
2ª sequência	+	3	3	7	5	2	3	3	7
		1	1	6	1	8	6	5	8	8

10. Dados dois números naturais m e n, com $1 \leqslant m, n \leqslant 100$, e duas sequências ordenadas com m e n números inteiros, obter uma única sequência ordenada contendo todos os elementos das sequências originais sem repetição.

Exercícios

11. Dada uma sequência de n números inteiros, com $1 \leqslant n \leqslant 100$, imprimi-la em ordem crescente de seus valores.
12. Dizemos que uma sequência de n elementos, com n par, é balanceada se as seguintes somas são todas iguais:
    a soma do maior elemento com o menor elemento;
    a soma do segundo maior elemento com o segundo menor elemento;
    a soma do terceiro maior elemento com o terceiro menor elemento;
    e assim por diante . . .
    Exemplo:
    $2\ 12\ 3\ 6\ 16\ 15$ é uma sequência balanceada,
    pois $16 + 2 = 15 + 3 = 12 + 6$.
    Dados n (n par e $0 \leqslant n \leqslant 100$) e uma sequência de n números inteiros, verificar se essa sequência é balanceada.

Exercícios

11. Dada uma sequência de n números inteiros, com $1 \leqslant n \leqslant 100$, imprimi-la em ordem crescente de seus valores.
12. Dizemos que uma sequência de n elementos, com n par, é balanceada se as seguintes somas são todas iguais:
    a soma do maior elemento com o menor elemento;
    a soma do segundo maior elemento com o segundo menor elemento;
    a soma do terceiro maior elemento com o terceiro menor elemento;
    e assim por diante . . .
    Exemplo:
    $2 \;\; 12 \;\; 3 \;\; 6 \;\; 16 \;\; 15$ é uma sequência balanceada,
    pois $16 + 2 = 15 + 3 = 12 + 6$.
    Dados n (n par e $0 \leqslant n \leqslant 100$) e uma sequência de n números inteiros, verificar se essa sequência é balanceada.

Exercícios

13. Dada uma sequência $x_1, x_2, \ldots, x_k$ de números inteiros, com $1 \leqslant k \leqslant 100$, verifique se existem dois segmentos consecutivos iguais nesta sequência, isto é, se existem i e m tais que

$$x_i, x_{i+1}, \ldots, x_{i+m-1} = x_{i+m}, x_{i+m+1}, \ldots, x_{i+2m-1}.$$

Imprima, caso existam, os valores de i e m.
Exemplo:
Na sequência $7, 9, \underline{5, 4}, \underline{5, 4}, 8, 6$ existem $i = 3$ e $m = 2$.

Exercícios

14. Dadas duas sequências de caracteres (uma contendo uma frase e outra contendo uma palavra), determine o número de vezes que a palavra ocorre na frase. Considere que essas sequências têm no máximo 100 caracteres cada uma.
    Exemplo:
    Para a palavra ANA e a frase:
    ANA E MARIANA GOSTAM DE BANANA.
    Temos que a palavra ocorre 4 vezes na frase.

Exercícios

15. São dadas as coordenadas reais x e y de um ponto, um número natural n e as coordenadas reais de n pontos, com $1 \leqslant n \leqslant 100$. Deseja-se calcular e imprimir sem repetição os raios das circunferências centradas no ponto (x, y) que passem por pelo menos um dos n pontos dados.
Exemplo:
$$\begin{cases} (x, y) = (1.0, 1.0) \\ n = 5 \\ \text{Pontos: } (-1.0, 1.2), (1.5, 2.0), (0.0, -2.0), (0.0, 0.5), (4.0, 2.0) \end{cases}$$
Nesse caso há três circunferências de raios $1.12, 2.01$ e 3.162.
Observações:
   - A distância entre os pontos (a, b) e (c, d) é $\sqrt{(a-c)^2 + (b-d)^2}$.
   - Dois pontos estão na mesma circunferência se estão à mesma distância do centro.