INSTITUTO FEDERAL
Sul-rio-grandense
Câmpus Passo Fundo

Ciência da
Computação

ALGORITMOS I

Professores:

Adilso Nunes de Souza

Maikon Cismoski dos Santos

ROTEIRO DA AULA

- Variáveis compostas
  - Conceitos
  - Definição e manipulação de array
  - Exemplo de teste de mesa

VARIÁVEIS COMPOSTAS HOMOGÊNEAS

- A construção de algoritmos que manipulam grandes volumes de dados, tendo a necessidade de armazenar cada um desses dados representa um problema significante para ser resolvido com variáveis literais que armazenam uma única informação a cada vez.

VARIÁVEIS COMPOSTAS HOMOGÊNEAS

- Exemplo de um problema:
  - Ler a idade de 50 pessoas e mostrar as idades em ordem crescente?
  - Como você resolveria isso?
  - Qual o custo operacional (tempo) para construir este algoritmo?
  - Se ampliar o número de pessoas para 5000?

VARIÁVEIS COMPOSTAS HOMOGÊNEAS

- Para resolver este tipo de problema podemos declarar variáveis que manipulam/armazenam mais do que uma informação.
- Quando uma variável é composta por vários valores do mesmo tipo primitivo, temos um conjunto homogêneo de dados, ou uma variável composta homogênea.

VARIÁVEIS COMPOSTAS HOMOGÊNEAS

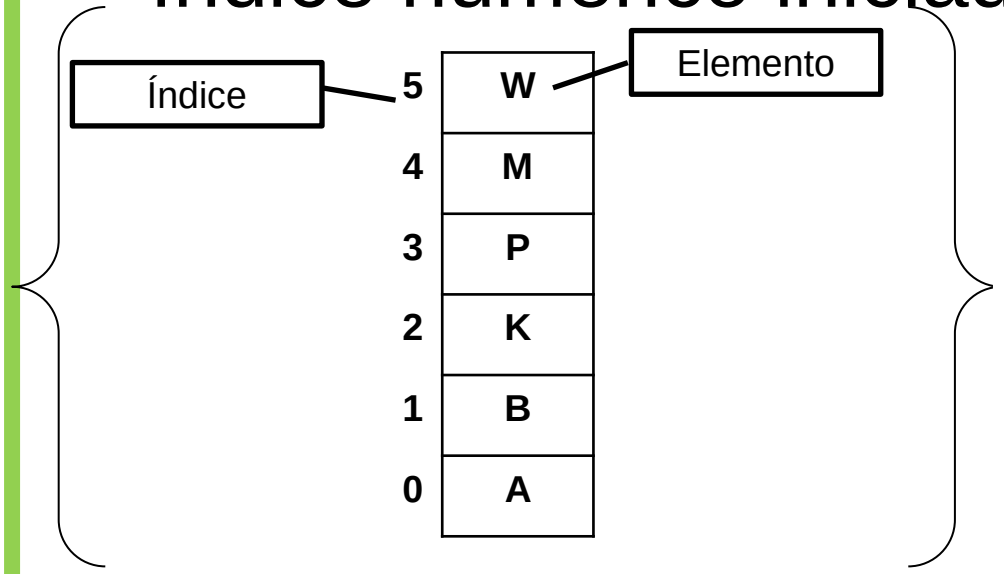
- Fazendo uma relação com o mundo real pode-se dizer que uma variável composta é uma alcateia e os valores que ela recebe são os lobos, então uma alcateia é formada por um conjunto de elementos da mesma espécie (os lobos).

VARIÁVEIS COMPOSTAS HOMOGÊNEAS

- As variáveis compostas podem ser:
  - **Unidimensionais**: variáveis indexadas com uma única dimensão, também conhecidas como **vetores ou array de uma dimensão**, são referenciadas por um único índice.
  - **Bidimensionais**: variáveis indexadas com duas dimensões, também conhecida como **matrizes ou array de duas dimensões**, são referenciadas por dois índices.

VARIÁVEIS INDEXADAS

- São chamadas de variáveis indexadas pois o acesso aos elementos é realizado através da utilização de um índice numérico iniciado em zero.



VETOR

- Sintaxe no portugol:
  - Declaração:

<identificador> : vetor [<inicio>..<fim>] de < tipo >
vet : vetor [0..9] de inteiro

  - No exemplo acima foi criado um variável dimensionada chamada “vet” com capacidade de armazenar 10 valores do tipo inteiro.

VETOR

- Para manipular a informação basta utilizar o nome da variável e entre colchetes o índice que deseja utilizar:
- Exemplo:

```
vet[0] <- 8
vet[3] <- 5 + 9
leia (vet[0])
escreva(vet[0])
```

- OBS: o índice será sempre um valor inteiro e poderá ser substituído por uma variável controladora.

VETOR EXEMPLO PORTUGOL

```
2  algoritmo "vetor1"
3
4  var
5     vet : vetor [0..9] de inteiro
6     x : inteiro
7
8  inicio
9     //leitura dos elementos do vetor
10    para x de 0 ate 9 faca
11       escreva("Digite o valor ", x, ": ")
12       leia (vet[x])
13    fimpara
14    limpatela
15
16    //mostra os elementos inseridos no vetor
17    para x de 0 ate 9 faca
18       escreval(vet[x])
19    fimpara
20 fimalgoritmo
```

VETOR EXEMPLO C++

```
 7  main()
 8  {
 9      int vet[10], i;
10      for(i = 0; i <= 9; i++)
11      {
12          cout << "Digite o valor " << i << ": ";
13          cin >> vet[i];
14          fflush(stdin);
15      }
16
17      for(i = 0; i <= 9; i++)
18      {
19          cout << vet[i] << ", ";
20      }
21  }
```

VETOR EXEMPLO C++

```
 7  main()
 8  {
 9      int vet[10], i = 0;
10      while(i <= 9)
11      {
12          cout << "Digite o valor " << i << ": ";
13          cin >> vet[i];
14          fflush(stdin);
15          i++;
16      }
17
18      i = 0;
19      while(i <= 9)
20      {
21          cout << vet[i] << ", ";
22          i++;
23      }
24  }
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

VETOR EXEMPLO C++

```
 7  main()
 8  {
 9      setlocale(LC_ALL, "Portuguese");
10      int vet[10], i = 0, soma = 0;
11      while(i <= 9)
12      {
13          cout << "Digite o valor " << i << ": ";
14          cin >> vet[i];
15          fflush(stdin);
16          soma += vet[i];
17          i++;
18      }
19
20      for(i = 0; i <= 9; i++)
21      {
22          cout << vet[i] << ", ";
23      }
24
25      cout << "\nSoma dos elementos do vetor: " << soma << endl;
26      cout << "\nMédia dos elementos do vetor: " << (float)soma / 10.0;
27  }
```

VALORES RANDOMICAMENTE

- Em muitos casos de teste a necessidade de entrada de um grande volume de dados pode ser uma tarefa dispendiosa e cansativa.
- Existe a possibilidade de gerar valores randomicamente para uma variável de qualquer tipo com determinados comandos aceitos pela linguagem C++.

VETOR RANDOMICAMENTE

- rand(): retorna um número inteiro pseudo-aleatório no intervalo entre 0 e RAND_MAX que é uma constante definida em <cstdlib>.
- Esse número é gerado por um algoritmo que retorna uma sequência de números aparentemente não relacionados cada vez que é chamado.
- Esse algoritmo usa uma semente para gerar a série, que deve ser inicializada com algum valor distinto usando a função srand.

VALORES RANDOMICAMENTE

```
1  #include<iostream>
2  #include<cstdio>
3  #include<cstdlib>
4  #include<ctime>
5
6  using namespace std;
7
8  main()
9  {
10     int vet[10], i;
11     srand(time(NULL)); //INICIALIZA O RANDÔMICO (SEMENTE)
12     for(i = 0; i <= 9; i++)
13     {
14         vet[i] = rand();
15     }
16
17     for(i = 0; i <= 9; i++)
18     {
19         cout << vet[i] << ", ";
20     }
21 }
```

VALORES RANDOMICAMENTE

- Uma maneira de gerar números pseudo-aleatórios em um determinado intervalo usando rand é usar o módulo do valor retornado pelo intervalo:

```
vet[0] = rand() % 100;
// valor no intervalo de 0 a 99
vet[0] = rand() % 100 + 1;
// valor no intervalo de 1 a 100
vet[0] = rand() % 30 + 1985;
// valor no intervalo 1985-2014
vet[3] = rand() % 3 - 1;
// valor no intervalo -1 até 2
```

VALORES RANDOMICAMENTE

```
 8  main()
 9  {
10      int vet[4], i;
11      srand(time(NULL)); //INICIALIZA O RANDÔMICO (SEMENTE)
12
13      vet[0] = rand() % 100;
14      // valor no intervalo de 0 a 99
15      vet[1] = rand() % 100 + 1;
16      // valor no intervalo de 1 a 100
17      vet[2] = rand() % 30 + 1985;
18      // valor no intervalo 1985-2014
19      vet[3] = rand() % 3 - 1;
20      // valor no intervalo -1 até 2
21
22      for(i = 0; i <= 3; i++)
23      {
24          cout << vet[i] << ", ";
25      }
26  }
```

VALORES RANDOMICAMENTE

- Gerar números float para preencher o vetor

```
 8  main()
 9  {
10      int i;
11      float vet[10];
12      srand(time(NULL)); //INICIALIZA O RANDÔMICO (SEMENTE)
13
14      for(i = 0; i <= 9; i++)
15      {
16          vet[i] = ((rand() % 100) / 100.0) * 50.0;
17      }
18
19      for(i = 0; i <= 9; i++)
20      {
21          cout << vet[i] << ", ";
22      }
23  }
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Passo Fundo

VALORES RANDOMICAMENTE

Linha	i	I <= 2	Vet[0]	Vet[1]	Vet[2]	soma
10	?	?	?	?	?	0
11	0	TRUE	?	?	?	
14			(5)	?	?	
16				?	?	5
17	1					
11		TRUE		?	?	
14				(4)	?	
16					?	9
17	2					
11		TRUE				
14					(2)	
16						11
17	3					
11		FALSE				
18						{11}

```
8   main()
9   {
10      int vet[3], i, soma = 0;
11      for(i = 0; i <= 2; i++)
12      {
13          cout << "Digite o valor " << i << ": "
14          cin >> vet[i];
15          fflush(stdin);
16          soma += vet[i];
17      }
18      cout << "Soma dos elementos do vetor: " <<
19  }
```

REFERÊNCIAS

- FORBELLONE, André Luiz Villar. Lógica de programação: a construção de algoritmos e estruturas de dados. 3 ed. São Paulo: Prentice Hall, 2005.
- VILARIN, Gilvan. Algoritmos Programação para Iniciantes. Editora Ciência Moderna. Rio de Janeiro, 2004.
- MORAES, Paulo Sérgio. Curso Básico de Lógica de Programação. Centro de Computação – Unicamp, 2000.
- STEINMETZ, Ernesto H. R.; FONTES, Roberto Duarte Cartilha Lógica de Programação. Editora IFB, Brasília - DF, 2013.