INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
CAMPUS NATAL - CENTRAL
DIRETORIA DE EDUCAÇÃO E TECNOLOGIA DA INFORMAÇÃO
Av. Sen. Salgado Filho, 1559, Natal/RN, 59015-000. Fone/FAX (084) 4005-2637
E-mail: dietinf@ifrn.edu.br - Site: http://www.ifrn.edu.br/dietinf

Curso: - Superior de Tecnologia em Análise e Desenvolvimento de Sistemas
- Superior de Tecnologia em Redes de Computadores

Disciplina: - Algoritmos e Técnicas de Programação
- Introdução à Programação

Professor: Fabiano Papaiz

ESTRUTURAS DE DADOS

Os tipos dados primitivos (inteiro, real, caractere e lógico) não são suficientes para representarmos todos os tipos de informação. Isso ocorre principalmente quando temos mais de uma informação relacionada, por exemplo: uma lista de nomes de alunos, um endereço etc.

Mas podemos construir novos tipos de dados a partir da composição de tipos primitivos. Estes novos tipos são denominados como estruturas de dados.

Vamos iniciar com falando sobre as estruturas de dados homogêneas, que são aquelas que permitem a manipulação de um conjunto de dados de um mesmo tipo. Primeiramente veremos a mais simples delas, o vetor. Depois aprenderemos uma estrutura um pouco mais complexa, a matriz. Por fim, falaremos sobre as cadeias de caracteres, mais conhecidas como strings.

Vetores

São conhecidos também por estruturas de dados homogêneas unidimensionais. Com eles, podemos criar um conjunto de dados (de um mesmo tipo), onde cada elemento do conjunto será identificado por um índice único.

Vamos imaginar um programa que irá receber uma série de 5 números e ao final exibirá a soma e a multiplicação destes números. Poderíamos logo imaginar: "basta criar 5 variáveis (*num1*, *num2, num3, num4* e *num5*), receber os números, calcular a soma e a multiplicação e exibir os resultados".

Tudo bem, isso irá funcionar. Mas, e se agora precisássemos receber 20 números ao invés de 5? Seguindo essa linha de pensamento, iríamos ter que manipular 20 variáveis (*num1* até *num20*) – e começaremos a perceber que a nossa programação irá ficar um pouco mais trabalhosa, pois criar 20 variáveis e manipulá-las corretamente levará um certo tempo de codificação e teste.

E se a quantidade de números aumentasse para 100? ou para 200?

Com esse programa, teríamos o trabalho de aumentar o número de variáveis a cada vez que a quantidade de números fosse aumentada. Quanto mais números, mais variáveis serão

necessárias e maior será o nosso tempo de codificação.

Vamos ilustrar como seria as duas soluções: sem utilizar um vetor e utilizando um vetor.

Sem Utilizar o Vetor

```
//variaveis para armazenar os numeros
//informados eo resultado da soma
int num1, num2, num3, num4, num5;
int soma, multiplicacao;
//obtem e armazena os numeros informados
cout << "Digite o 1o. numero" << endl;
cin >> num1;
cout << "Digite o 2o. numero" << endl;
cin >> num2;
cout << "Digite o 3o. numero" << endl;
cin >> num3;
cout << "Digite o 4o. numero" << endl;
cin >> num4;
cout << "Digite o 5o. numero" << endl;
cin >> num5;
//processa a soma
soma = num1+num2+num3+num4+num5;
//processa a multiplicacao
multiplicacao = num1*num2*num3*num4*num5;
//exibe o resultado
cout << "SOMA = " << soma << endl;
cout << "MULTIPLICACAO = " << multiplicacao<< endl;
system("pause");
```

Utilizando o Vetor

```
//vetor para armazenar os 5 numeros
int numeros[5];
int soma, multiplicacao;
//obtem e armazena os numeros informados
for (int i=1; i<=5; i++){
    cout << "Digite o " << i << "o. numero" << endl;
    cin >> numeros[i-1];
}
//processa a soma e a multiplicacao
soma = 0;
multiplicacao = 1;
for (int i=0; i<5; i++){
    soma += numeros[i];
    multiplicacao *= numeros[i];
}
//exibe o resultado
cout << "SOMA = " << soma << endl;
cout << "MULTIPLICACAO = " << multiplicacao<< endl;
system("pause");
```

Analisando o código:

```
//vetor para armazenar 5 numeros
int numeros[5];
```

Essa instrução irá criar uma variável chamada numeros que será um vetor para 5 elementos do tipo int. A quantidade de elementos foi definida dentro dos [ ].

Os elementos de um vetor podem ser acessados a partir de seu índice e usados no programa como qualquer outra variável. Os índices de um vetor de N elementos variam de 0 a N-1 (onde 0 corresponde ao primeiro elemento e N-1 corresponde ao último). Portanto, o primeiro elemento do vetor numeros estará no índice 0 e o último no índice 4 (*5-1*).

Abaixo representamos o vetor numeros caso este fosse formado pelo conjunto {5, 14, 3, 20 e 1}.

5	14	3	20	1
Índice 0	*Índice 1*	*Índice 2*	*Índice 3*	*Índice 4*
numeros[0]	numeros[1]	numeros[2]	numeros[3]	numeros[4]

Portanto, o 1º elemento estará sempre na posição 0 (zero) e o último na posição N-1 (onde N é a quantidade de elementos do vetor).

Qual das opções abaixo irá obter o valor do 3º elemento de numeros?

a) `numeros[1]` b) `numeros[3]` c) `numeros[2]`

Podemos atribuir valores para os elementos de um vetor de duas formas: acessando cada elemento pelo seu índice ou utilizar um conjunto de valores entre { }. No exemplo a seguir, os vetores vetor1 e vetor2 serão equivalentes:

```
int vetor1[3];
vetor1[0] = 1;
vetor1[1] = 2;
vetor1[2] = 3;

int vetor2[3] = {1, 2, 3};
```

Vamos continuar analisando o código:

```
//obtem e armazena os numeros informados
for (int i=1; i<=5; i++){
    cout << "Digite o " << i << "o. numero" << endl;
    cin >> numeros[i-1];
}
```

Nesta parte, fazemos um *loop* FOR para que o usuário informe os 5 números. Observe que armazenamos cada número informado no seu índice correspondente do vetor numeros - através da instrução `numeros[i-1]`.

```
//processa a soma e a multiplicacao
soma = 0;
multiplicacao = 1;
for (int i=0; i<5; i++){
    soma += numeros[i];
    multiplicacao *= numeros[i];
}
```

Aqui inicializamos as variáveis soma e multiplicacao e a seguir fazemos um novo loop FOR para obtermos o valor de cada elemento do vetor numeros. A cada iteração acumulamos a soma e a multiplicação dos elementos.

Exercícios:

1. Altere o programa de exemplo para que agora ele possa receber 15 números em vez de 5.
2. Altere novamente o programa de exemplo para que agora ele possa receber 7 números.
3. Crie um vetor de 50 elementos. Em seguida, utilizando um *loop* FOR, você deverá fazer com que o valor de cada elemento seja igual ao triplo do valor do seu índice. Por último, faça um outro *loop* FOR para exibir os valores dos elementos do vetor. Exemplo de como serão os valores dos elementos:

```
vetor[0] = 0    (3 x 0)
vetor[1] = 3    (3 x 1)
vetor[2] = 6    (3 x 2)
vetor[3] = 9    (3 x 3)
  :
```