Vetores em C e *C*++

(parte 2)

Usando vetores com funções

CI208/UFPR: Programação de Computadores

Prof. Wagner M. Nunan Zola

Setembro/2020

Usando vetores com funções

- Uma maneira simples de usar funções com vetores está em aplicar funções a células de um vetor e, por exemplo produzir outros vetores.

- Por exemplo, o programa a seguir processa um vetor de valores de A (de ângulos em radianos) e produz outro vetor de floats chamado CosA.

- O vetor CosA possui os cossenos dos ângulos previamente calculados e armazenados dos ângulos em posições correspondentes do vetor A.

Programa exemplo para processar um vetor e produzir outro com função

```
/* Uma maneira simples de usar funções com vetores está em aplicar funções a células de um vetor e, por
exemplo produzir outros vetores. Ler o vetor de N ângulos em radianos (N definido por #define). A seguir
processar um vetor A e produzir outro vetor de floats chamado CosA. Imprimir o vetor CosA. */
#include <iostream>
#include <cmath>
using namespace std;

#define N 20

int main ( ) {
    float A[ N ], CosA[ N ];        // declara os 3 vetores
    int i;

    cout << "Digite o vetor A (" << N << " angulos em radianos): ";
    for( i=0; i < N; i++ )
       cin >> A[i];

    for( i=0; i < N; i++ )    // processa o vetor A produzindo o vetor CosA
       CosA[i] = cos( A[i] );

    cout << "O vetor de cossenos é: ";
    for( i=0; i < N; i++ )            // imprime o vetor CosA
       cout << CosA[i] << " ";
    cout << endl;

   return 0;

}  // fim do programa
```

Alterando células de vetores com funções: passando parâmetros por referência

- Outra maneira simples de usar funções com vetores está em aplicar funções a células de um vetor e **alterar as células do próprio vetor**.

- Podemos fazer isso com passagem de parâmetros por referência (passando células do vetor por referência).

- Por exemplo vamos usar a função `troca` (já vista) para refazer o programa de reversão de ordem de uma sequência (exemplo da parte 1 dos slides), porém faremos isso **sem usar um vetor auxiliar.**

- Veremos que o programa começa a ficar mais fácil de ser lido (entendido) desde que saibamos o mecanismo da passagem por referência

Relembrando o exemplo da parte 1: ler, copiar em ordem reversa, imprimir vetor

```
#include <iostream>          // Fazer um programa para ler sequencia de N inteiros armazenando no vetor Vet.
using namespace std;         // Copiar o vetor Vet em OUTRO vetor Rev em ordem reversa. Imprimir o vetor Rev

#define N 10                                  // N definido por #define

int main ( ) {

    int Vet[ N ];         // declara um vetor de N ints
    int Rev[ N ];         // declara um vetor de N ints para conter a cópia reversa
    int i;

    cout << "Digite uma sequencia de " << N << " inteiros: ";
    for( i=0; i < N; i++ )
        cin >> Vet[i];

    for( i=0; i < N; i++ )          // percorre o vetor TODO em ordem, MAS ...
        Rev[N-1-i] = Vet[i];        //   copia para Rev nas posições (*) em ordem reversa
                                    // OBS: (*) ALTERADO em relação à parte 1 dos slides  (era 9),
                                    //                para usar o valor N definido -1

    cout << "O vetor Rev contém: ";
    for( i=0; i < N; i++ )
        cout << Rev[i] << " ";
    cout << endl;                   // troca de linha ...
                                    //  ... depois de imprimir a sequência

   return 0;
}  // fim do programa
```

Compilando com **N par (exemplo: #define N 10).** Funciona!

```
Digite uma sequencia de 10 inteiros: 1 2 3 4 5 6 7 8 9 10
O vetor Rev contém: 10 9 8 7 6 5 4 3 2 1
```

Compilando com **N impar (exemplo: #define N 9).** Funciona!

```
Digite uma sequencia de 9 inteiros: 1 2 3 4 5 6 7 8 9
O vetor Rev contém: 9 8 7 6 5 4 3 2 1
```

Refazendo o exemplo anterior usando função troca (passagem por referência)

```
#include <iostream>          // Fazer um programa para ler sequencia de N inteiros armazenando no vetor Vet.
using namespace std;         // Reverter a ordem do vetor Vet (sem usar vetor extra/auxiliar). Imprimir o vetor
#define N 10                 // N definido por #define

void troca( int &a, int &b ) {
   int copia;   // variável auxiliar (mantém copia de a)

   copia = a;
   a = b;
   b = copia;
}

int main ( ) {
    int Vet[ N ];        // declara um vetor de N ints
    int i;

   cout << "Digite uma sequencia de " << N << " inteiros: ";
   for( i=0; i < N; i++ )              // lê o vetor
       cin >> Vet[i];

    for( i=0; i < N/2; i++ )        // vai até a metade (ou quase metade, se N for impar) do vetor, MAS ...
       troca( Vet[i], Vet[N-1-i] );          //  … troca com sua posição simétrica

    cout << "O vetor Vet contém: ";
    for( i=0; i < N; i++ )              // imprime o vetor resultante
       cout << Vet[i] << " ";
    cout << endl;             // troca de linha depois de imprimir a sequência

    return 0;
}  // fim do programa
```

Compilando com **N par (exemplo: #define N 10).** Funciona!

```
Digite uma sequencia de 10 inteiros: 1 2 3 4 5 6 7 8 9 10
O vetor Vet contém: 10 9 8 7 6 5 4 3 2 1
```

Compilando com **N impar (exemplo: #define N 9).** Funciona!

```
Digite uma sequencia de 9 inteiros: 1 2 3 4 5 6 7 8 9
O vetor Vet contém: 9 8 7 6 5 4 3 2 1
```

OBS: Funciona mesmo percorrendo até a posição 3 com N=9 impar.
Note que percorre de 0 enquanto i < 9/2 (ou seja, de 0 a 3, pois não é necessário trocar a posição do meio com N impar.)

Passando todo o vetor como parâmetro de funções

- Veremos agora mais um caso de vetores e funções. Vamos passar TODO o vetor como parâmetro.

- Na declaração da função, podemos deixar o tamanho **do vetor** sem especificação. Basta escrever **[]** depois do nome do vetor passado como parâmetro (colchetes vazios). Dessa forma, vamos na direção de uma função mais genérica que **pode funcionar com vetores de qualquer tamanho**.
  - No entanto *necessitamos passar MAIS UMA VARIÁVEL como parâmetro*. Essa variável deve indicar o número de elementos do vetor que serão considerados durante a execução do programa.

- Podemos fazer , por exemplo uma função para ler vetor, outra para imprimir vetor, outra para somar vetores, etc.

- No **exemplo a seguir** vamos fazer um programa para ler dois vetores e fazer a soma vetorial dos mesmos com resultado em um terceiro vetor

Agora podemos reescrever de soma vetorial com funções

```
#include <iostream>     //  Fazer um programa para ler dois vetores A e B, de até 500 elementos float.
using namespace std;    // O programa deve perguntar com quantos elementos se deseja trabalhar no vetor.
                        // O programa deve calcular o vetor C com a soma vetorial:  C = A + B.
                        // Em seguida o programa deve imprimir o vetor C resultante.

#define N 500                   // N máximo definido por #define

// Declarando funções a serem usadas em main

void leVet( float V[ ], int n ) {
    for( int i=0; i < n; i++ )
        cin >> V[i];
}

void imprimeVet( float V[ ], int n ) {
    for( int i=0; i < n; i++ )
        cout << V[i] << " ";
}

 // função para calcular a soma vetorial: Vc = Va + Vb
void somaVet( float Vc[ ], float Va[ ], float Vb[ ], int n ) {
    for( int i=0; i < n; i++ )
        Vc[i] = Va[i] + Vb[i];
}

    // continua
```

```
int main ( ) {
    float A[ N ], B[ N ], C[ N ];      // declara  os 3 vetores
    int n;       // quantidade de posições a serem
                 //  usadas nos vetores

     cout << "Quantas posições em cada vetor? ";
     cin >> n;
     if( n > N ) {
        cout << "Esse programa foi compilado"
             << "para o máximo de " << N
             << " posições em cada vetor ! " << endl;
        return 0; // termina programa
      }

      // usa as funções declaradas para fazer
      //    ...    a tarefa em n posições de cada
      //    …   vetor (CONTINUA no slide a seguir)

      return 0;

  }  // fim do programa
```

Agora podemos reescrever de soma vetorial com funções

```
#include <iostream>     //  Fazer um programa para ler dois vetores A e B, de até 500 elementos float.
using namespace std;    // O programa deve perguntar com quantos elementos se deseja trabalhar no vetor.
                        // O programa deve calcular o vetor C com a soma vetorial:  C = A + B.
                        // Em seguida o programa deve imprimir o vetor C resultante.

#define N 500                   // N máximo definido por #define

// Declarar aqui em … funções a serem usadas em main
//  void leVet( float V[], int n ) { … }
//  void imprimeVet( float V[], int n ) { … }
//  void somaVet( float Vc[], float Va[], float Vb[], int n ) { … }

int main ( ) {
    float A[ N ], B[ N ], C[ N ];      // declara  os 3 vetores
    int n;       // quantidade de posições a serem
                 //  usadas nos vetores

     cout << "Quantas posições em cada vetor? ";
     cin >> n;
     if( n > N ) {
        cout << "Esse programa foi compilado"
             << "para o máximo de " << N
             << " posições em cada vetor? " << endl;
        return 0; // termina programa
      }

    // continua
```

```
// usa as funções declaradas para fazer a tarefa
//  ... em n posições de cada vetor
// CONTINUAÇÃO do slide anterior

cout << " Digite o vetor A: ";
leVet( A, n );
cout << " Digite o vetor B: ";
leVet( B, n );

somaVet( C, A, B, n );

cout << "vetor C somado: ";
imprimeVet( C, n );


return 0;

}  // fim do programa
```

Executando a soma vetorial com funções

```
Quantas posições em cada vetor? 5
Digite o vetor A: 1 1 2 2 3
Digite o vetor B: 1 1 2 2 3
vetor C somado: 2 2 4 4 6
```

Executando novamente:

```
Quantas posições em cada vetor? 7
Digite o vetor A: 1 1 2 2 3 1 1
Digite o vetor B: 1 1 2 2 3 1 1
vetor C somado: 2 2 4 4 6 2 2
```

Exercícios adicionais: repita os exercícios da parte 1, desta vez use funções+vetores para encapsular as etapas do seus programas

1) Faça um programa que leia DOIS vetores (**A** e **B**) de **N** posições com números reais (**N** definido por #define). O programa deve calcular e imprimir o produto escalar dos dois vetores.

2) Faça um programa que usa um vetor **Vet** de até **N** posições com inteiros positivos (**N** definido por #define). O programa deve ler os números e colocá-los no vetor **Vet**. O último número digitado será negativo e apenas indicará o fim da digitação do vetor **Vet**.

O programa deve contar quantos inteiros positivos foram colocados no vetor.

Em seguida o programa deve ler números do teclado e para cada número lido indicar ao usuário se ele está ou não no vetor **Vet**. Nesse ponto, a digitação de um número negativo deve causar o término do programa.

3) Faça um programa que leia DOIS vetores (**A** e **B**) de **N** posições com números inteiros (**N** definido por #define). Suponha que **A** e **B** representam conjuntos e seus elementos estão em ordem qualquer.

O programa deve calcular e imprimir o vetor C tal que $\mathbf{C = A \cap B}$,

ou seja **C** deve conter a interseção entre os conjuntos **A** e **B**.

Bibliografia adicional:
favor consultar também

“Linguagem C++ - Notas de Aula”. Revisão para C++ por Armando Luiz N. Delgado, a partir de material de Carmem S. Hara e Wagner N. Zola. 2018.
http://www.inf.ufpr.br/ci208/NotasAula/notas-13_Vetores_ou_Arrays.html#SECTION010137000000000000000

Créditos: O conteúdo original deste documento é de autoria do Prof. Wagner M. Nunan Zola, para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183)

Compartilhe este documento de acordo com a licença abaixo