Fatec Taubaté

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

ESTRUTURA DE DADOS

Revisão de Vetores, Ponteiros e Modularização

profa. Divani Barbosa Gavinier

divani.gavinier@fatec.sp.gov.br

1

Dia	Aula
7-8/ago	Apresentação do Plano de Aula / Introdução a Linguagem de Programação C++
14-15/ago	Revisão de Vetores, Ponteiros e Modularização
21-22/ago	Objetos e Classes
28-29/ago	Vetores Ordenados, Busca Binária e Notação do O grande

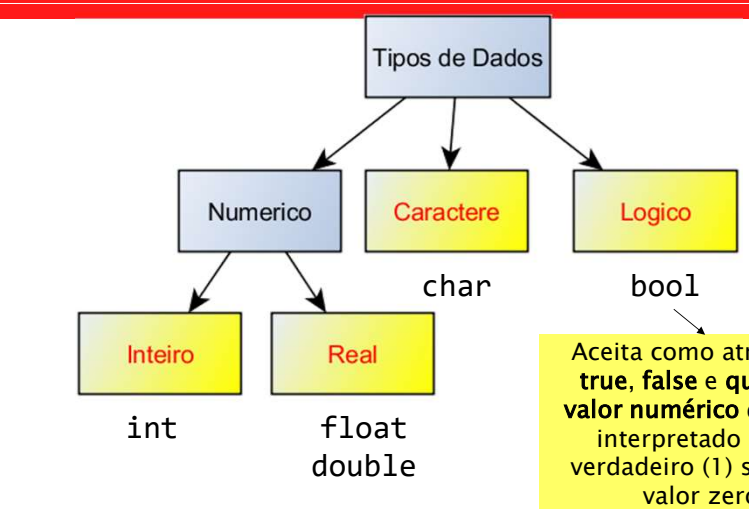
07 - Início das aulas do 2º semestre letivo de 2023

15 - Prazo final de alterações de matrículas para acomodação de horários - (Art. 33 Regulamento)

2

2

Tipos de Dados Primitivos



Possui um único valor !!!

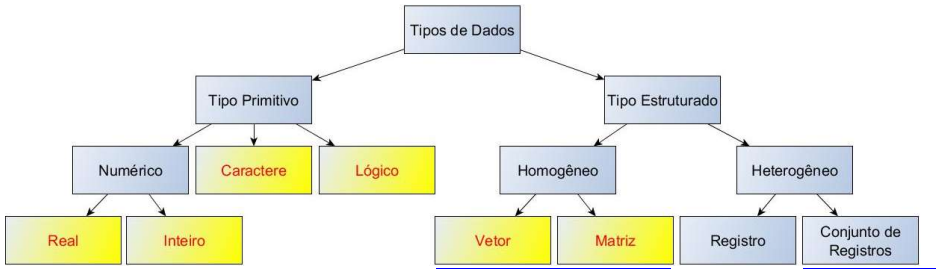
Aceita como atribuição **true**, **false** e **qualquer valor numérico** que será interpretado como verdadeiro (1) se não o valor zero.

Saída na tela é 1 ou 0.

3

3

Tipos de Dados Estruturados



Conteúdo não pode ser decomposto, ou seja, possui um único valor!!!

Tipo Primitivo Igual

Diferentes Tipos Primitivos

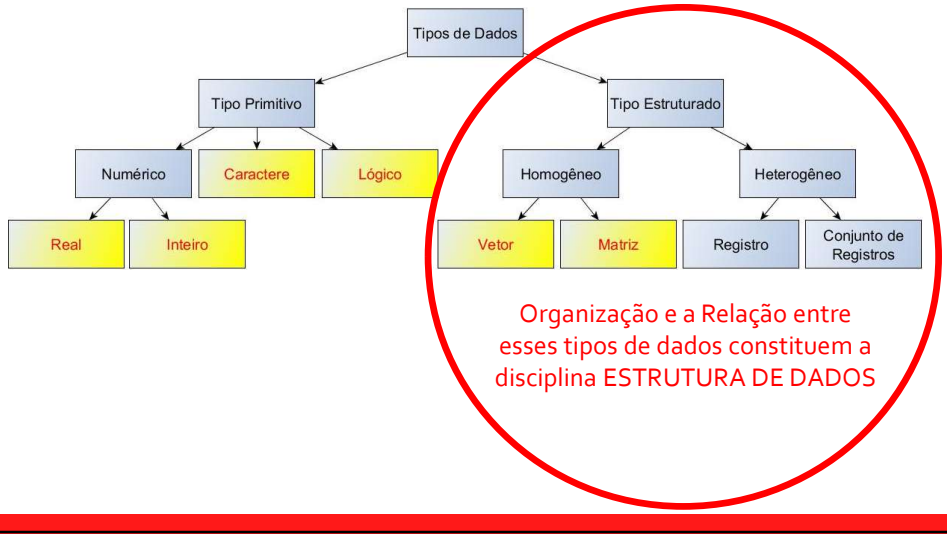
Diferentes Tipos Primitivos e Estruturado

Possuem o conteúdo composto de valores distintos apesar de possuir um único identificador (nome) !!!

4

4

Tipos de Dados Estruturados



Organização e a Relação entre esses tipos de dados constituem a disciplina ESTRUTURA DE DADOS

5

5

Arranjos

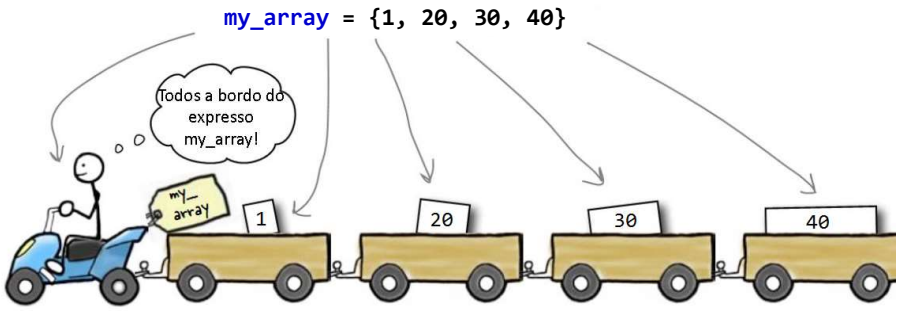
É uma variável que representa um conjunto de variáveis do mesmo tipo, onde cada uma pode armazenar um conteúdo diferente, mas que compartilham o mesmo identificador (nome).

6

6

Arranjos

É uma variável que representa um conjunto de variáveis do mesmo tipo, onde cada uma pode armazenar um conteúdo diferente, mas que compartilham o mesmo identificador (nome).



O trem my_array é uma única variável

7

7

Vídeo (12 min)

https://youtu.be/NvSLfK-JSo4

8

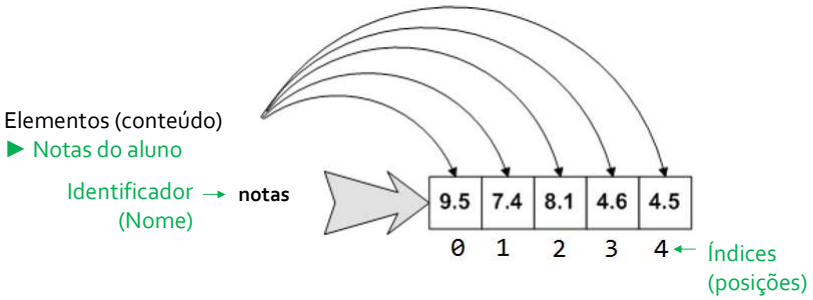
8

Vetores

Estrutura de armazenamento de dados mais comumente usado.

Vetores em C++ são iguais ao C

Exemplo: Vetor do tipo float com notas de 5 alunos



9

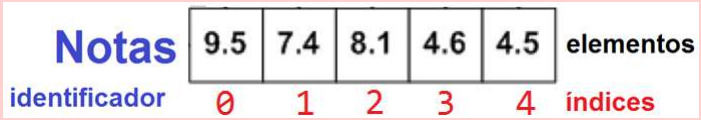
9

Declaração

A declaração de um vetor em C++ é igual em C, segue a seguinte sintaxe:

```
tipo <nome_variavel>[<numero de elementos>];
```

Atividade 1: Declare o vetor abaixo:



Resposta:

```
float Notas[5];
```

Atividade 2: Declare um vetor de nome v, do tipo inteiro com no máximo 100 elementos:

Resposta:

```
int v[100];
```

10

10

Inserção

Inserir um item no vetor é fácil, usamos a sintaxe normal de vetor e também controlamos quantos itens inserimos no vetor com a variável n.

```
int n;
v[0] = 77;
v[1] = 55;
n = 2;
```

Exibição

Considerando que a variável **n** contem o número de elementos do vetor. Exibir todos os elementos é simples, percorremos o vetor acessando cada elemento com `v[i]` e exibimo-los.

```
int i; // contador para varrer os índices de cada elemento
for(i=0; i<n; i++) // para cada elemento
   cout << " " << v[i]; // mostrar item
cout << endl; // pulando uma linha após impressão
```

11

11

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 1 (chave no vetor)

n = ?

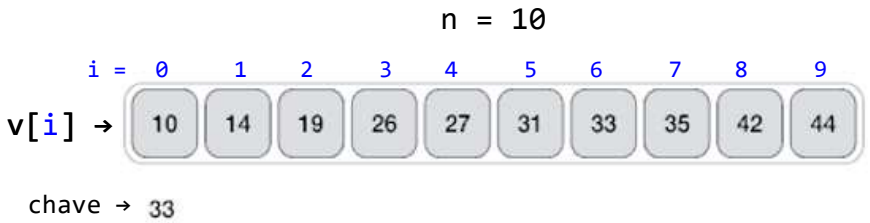
chave → 33

12

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 1

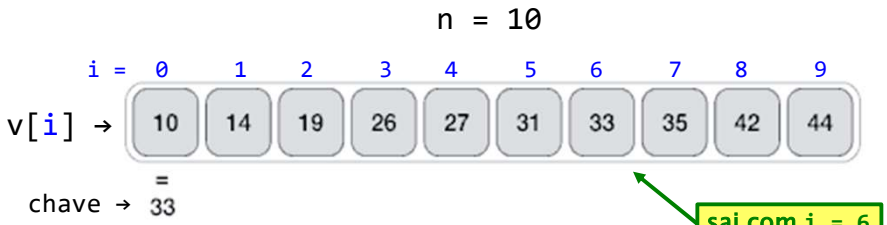


13

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 1




```
int chave = 33; // achar item com chave 33
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

14

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2 (chave não se encontra no vetor)

n = 10


```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

15

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2

n = 10

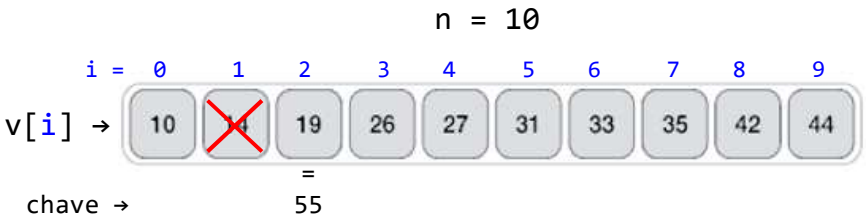

```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

16

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2



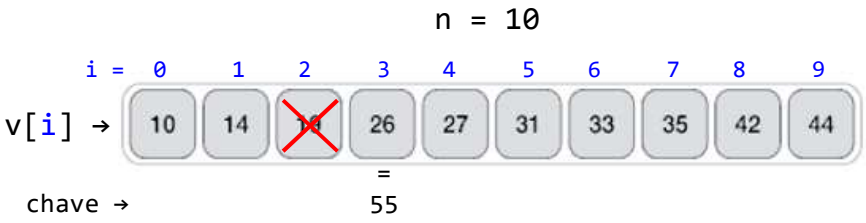

```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

17

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2



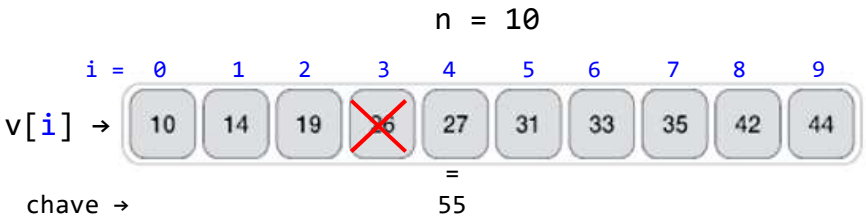

```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

18

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2



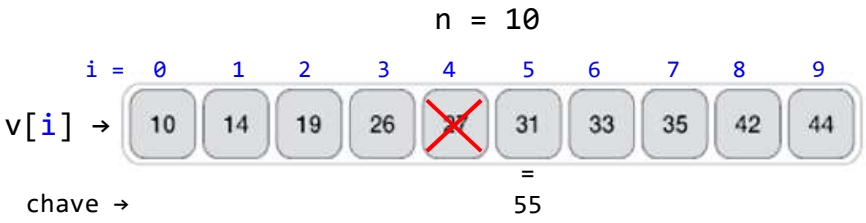

```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

19

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2




```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

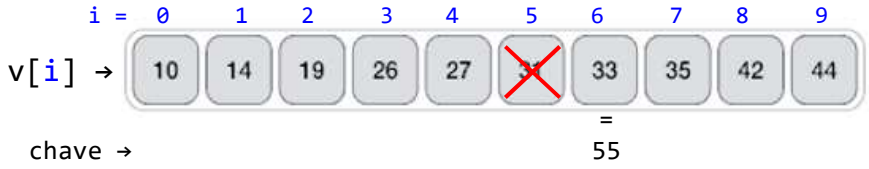
20

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2

n = 10




```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

21

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2

n = 10


```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

22

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2

n = 10


```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

23

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2

n = 10

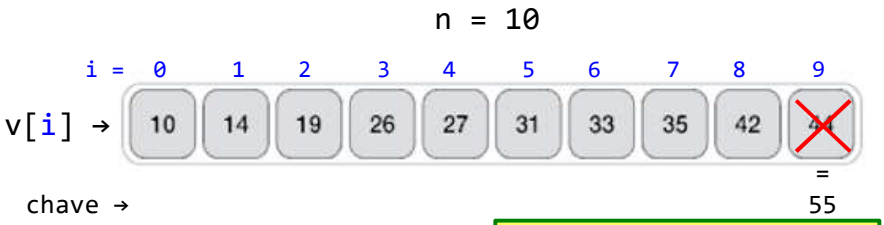

```
int chave = 55; // achar item com chave 55
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

24

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Exemplo 2




```
int chave = 55; // achar item com
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for
}
```

sai com i = 10 ou seja i = n

25

Pesquisa

Considerando que a variável chave contem o valor que estamos procurando. Para pesquisar um item, percorremos o vetor comparando a chave com cada elemento. Se o contador i atingir a ultima célula ocupada sem nenhuma coincidência encontrada, o valor procurado não se encontra no vetor.

Trecho de código completo para pesquisa

```
int chave = 55; // achar item com chave 55

for (i=0; i<n; i++) // para cada elemento
    if ( v[i] == chave ) // achou item?
       break; // sim achei, saída imediata do laço for

if (i == n) // chegou ao fim do laço for sem encontrar?
  cout << "Chave NAO encontrada" << endl; // sim
else
  cout << "Chave encontrada no indice " << i << endl; // não
```

26

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 1 (chave no final) n = 10

27

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

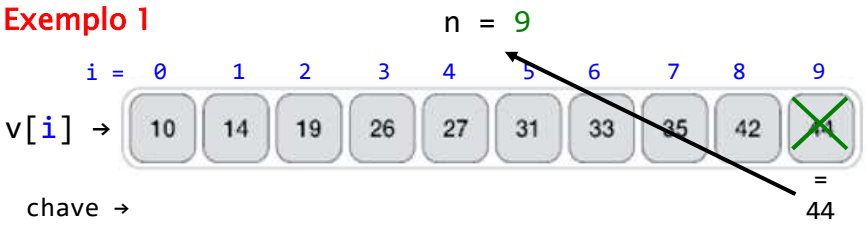
Exemplo 1 n = 10

28

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 1




```
int chave = 44; // achar item com chave 44
for (i=0; i<n; i++) { // para cada elemento
    if ( v[i] == chave ) { // achou item?
       n--; // sim achei, decrementa um item do vetor
       break; // sim achei, saída imediata do laço for
    }
}
```

29

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 2 (chave não no final)

30

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 2

n = 10

Vetor v após a remoção deve ficar dessa forma:

n = 9

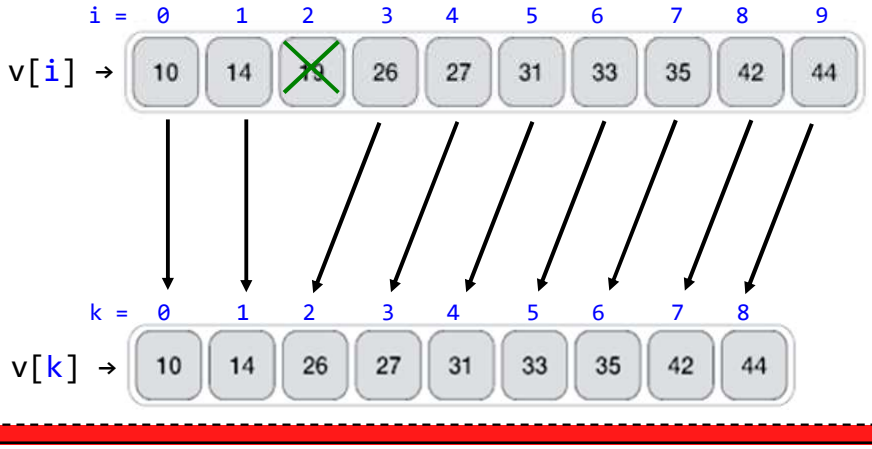
31

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 2

n = 10

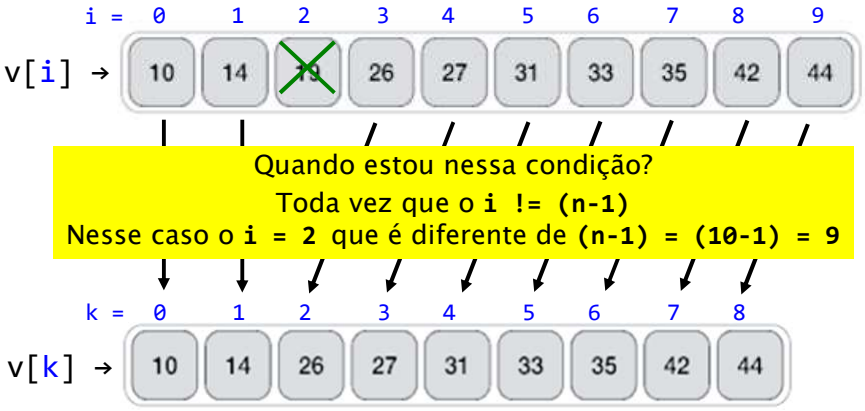


32

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 2 n = 10

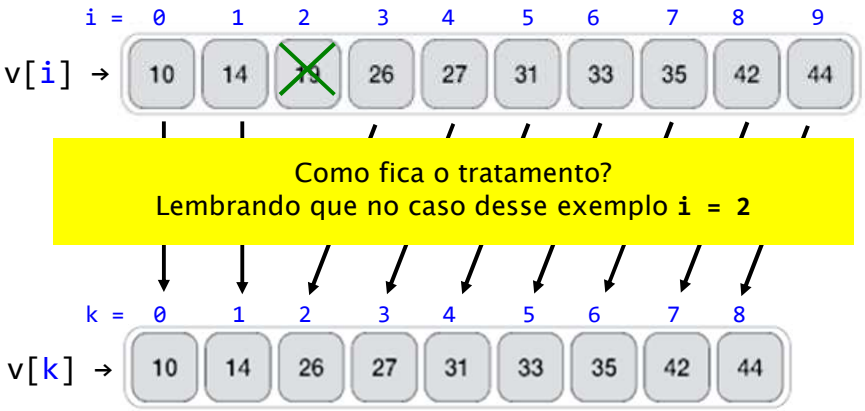


33

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 2 n = 10

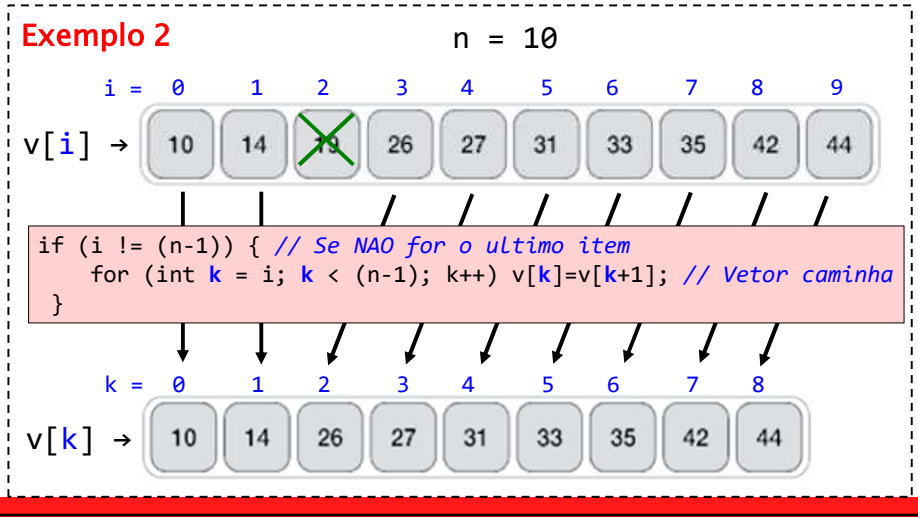


34

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Exemplo 2



35

Remoção

A remoção começa com a pesquisa do item especificado (chave). Caso ele se encontre no vetor movemos todos os itens de índice mais alto para baixo e diminuímos o numero de elementos.

Trecho de código completo para remoção

```
chave = 19; // remover item com chave 19
for (i=0; i<n ;i++) { // consulta de cada elemento

   if (v[i] == chave) { // achou item?

      if (i != (n-1)) { // Se NAO for o ultimo item
         for(int k=i; k<(n-1); k++) // Vetor caminha
              v[k]=v[k+1];
      }

      n--; // decrementa um elemento no vetor
      break; // achei e removi item, sair antes do fim
   } // fim if (achou o item)

 } // fim laço for (consulta de cada elemento)
```

36

Programa VetorApp.cpp

VetorApp.cpp

```
#include <iostream>

using namespace std;

main() {

 int v[100], n, i; // declaração das variáveis a serem usadas

 cout << "Inserindo 10 elementos no vetor" << endl;
 v[0] = 10;  v[1] = 14;  v[2]=19;  v[3]=26;  v[4]=27;
 v[5] = 31;  v[6] = 33;  v[7]=35;  v[8]=42;  v[9]=44;
 n = 10;

 cout << "\nMostrando todos elementos do vetor" << endl;
 for(i=0; i<n; i++)
   cout << " " << v[i];
 cout << endl;

                                                  // Continua ...
```

37

VetorApp.cpp

```
// Continuação ...

 cout << "\nProcurando item 55" << endl;
 int chave = 55;
 for (i=0; i<n; i++)
    if ( v[i] == chave )
       break;
 if (i == n)
    cout << "Chave NAO encontrada" << endl;
 else
    cout << "Chave encontrada no indice " << i << endl;

 cout << "\nRemovendo item 19" << endl;
 chave = 19;
 for (i=0; i<n ;i++) {
    if (v[i] == chave) {
       if (i != (n-1)) {
          for(int k=i; k<(n-1); k++)  v[k]=v[k+1];
       }
       n--;
       break;
    }
 }
                                                  // Continua ...
```

38

VetorApp.cpp

```
// Continuação ...

 cout << "\nMostrando todos elementos do vetor" << endl;
 for(i=0; i<n; i++)
   cout << " " << v[i];
 cout << endl;

} // fim do programa principal (main)
```

39

AVISO IMPORTANTE

É de extrema importância que além desse material de apoio, tenham acesso a sites que possam ajudá-los em seus aprimoramentos estudantis (como sites de busca e de vídeos) e que consultem o ambiente do curso no SIGA para terem mais informações sobre os conteúdos estudados.

40

40

Textos de Apoio

Para que você possa melhorar seus estudos, consulte também os seguintes sites:

- **Programar em C+/Vetores.** Disponível em: https://pt.wikibooks.org/wiki/Programar_em_C%2B%2B/Vetores Acesso em: fevereiro 2022

- **Programas exemplos de Matrizes e Vetores em C.** Disponível em: https://www.inf.pucrs.br/~pinho/LaproI/Vetores/Vetores.htm#ExemplosVet Acesso em: fevereiro 2022

- **Notas de aula em pdf: Vetores em C.** Disponível em: http://www3.decom.ufop.br/toffolo/media/uploads/2020-3_ple-e/bcc201/15_vetores.pdf Acesso em: fevereiro 2022

41

41

Vídeo Aulas

Para que você possa melhorar seus estudos, consulte também as vídeos aulas:

- **Curso de C++ - Aula 15 - Vetores** (10 min) Disponível em: https://www.youtube.com/watch?v=cdEccuCz34w Acesso em: fevereiro 2022

- **C++ - Vetores #19** (7 min) Disponível em: https://youtu.be/JO9VWtLaFpA Acesso em: fevereiro 2022

- **Linguagem Vetores** (12 min) Disponível em: https://youtu.be/NvSLfK-JSo4 Acesso em: fevereiro 2022

42

42

Questionário Online

O questionário da aula de hoje, , encontra-se no link: https://forms.gle/BJbAKyhyHf2afRc77

Para ter acesso ao questionário se faz necessário o uso da conta institucional @fatec.sp.gov.br.

Escrever programas para resolver projetos de programação ajuda a solidificar a compreensão da aula e demonstra como os conceitos apresentados na aula podem ser aplicados

O questionário online é auto avaliativo, ou seja, as resposta das questões serão apresentadas logo após o envio do formulário.

43

Fim
Aula de Introdução

44

44

Ir além é tão incerto quanto não alcançar o objetivo. Confúcio.

VETORES

Um vetor é um grupo de variáveis (elementos ou componentes) que contém valores do mesmo tipo, isto é, se for um vetor de inteiros, todos os elementos serão números inteiros. Ao se declarar um vetor é reservado um número finito de espaços de memória (caixinhas), de tamanho compatível com o tipo vetor. A declaração de um vetor em C segue a seguinte sintaxe:

```
tipo nome_variavel [no. de elementos]
```

onde:
tipo – corresponde ao tipo de dados de cada elementos do vetor.
nome_variavel – indica por qual nome este vetor será conhecido.
No. de elementos – valor constante que indica quantos elementos o vetor tem.

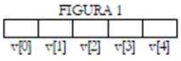
É importante relembrar que o vetor pode conter elementos de qualquer tipo de dados (por ex: float, integer, char). No entanto, todos os elementos do vetor têm o mesmo tipo que foi utilizado na declaração do vetor.

PT1: Neste exemplo é declarado um **vetor**, de inteiros, de tamanho 5 e um outro **vetor** capaz de armazenar 10 números reais.

```
int   v[5];
float s[10];
```

O resultado da declaração dos vetores v e s no **PT1** é tal que um espaço de memória (como se fosse uma caixa) é reservado para receber valores de acordo com o tipo declarado para cada vetor. No caso do vetor v tem-se uma caixa v, com 5 divisões, como mostra a figura 1 abaixo. Essas divisões são indexadas por números, por exemplo, v[0] é a primeira divisão, v[1] é a segunda e assim sucessivamente. Esses números são os índices do vetor.

FIGURA 1



A partir da Figura 1 é possível verificar que os índices de um vetor com **n** elementos variam entre **0** e **n-1**. Ou seja, o primeiro elemento, de qualquer vetor em C, é sempre zero e o n-ésimo elemento está sempre na posição **n-1**. Além disso, cada uma das 5 posições do vetor é acessada através do respectivo índice colocado entre **Colchetes [ ]**.

PT2: Termine de construir o código abaixo para inicializar e imprimir os elementos do **vetor S** com a seqüência {1,2,3,4,5,0,0,0,0,0}.

```
#include <stdio.h>
main()
{
  int i;
  int v[5];
  float S[10];
  for (i=0; i< 5; i++)
  {
    v[i] = i + 1;
  }
  printf("v[i] = ");
  for (i=0; i< 5; i++)
  {
    printf("%d",v[i]);
  }

}
```

PE1: Modifique o **PT2** para que o usuário possa entrar com os 5 valores de v através da função **scanf**. e o vetor S receba os 5 primeiros valores do vetor v e os demais seceba zero.

PE2: Modifique o **PE1** de modo a imprimir os elementos do vetor em uma linha e na linha logo abaixo os índices do vetor tal como mostrado na **Figura 1**.

INICIALIZAÇÃO DE VETORES

Tal como as variáveis, os vetores quando são criados contêm valores aleatórios (como se fosse um tipo de lixo) em cada uma das suas posições. No entanto pode-se atribuir valores ao vetor (ser inicializado) no momento da declaração. Nesse caso se o vetor for declarado com **n** elementos e ele for inicializado com apenas **k** valores ($k < n$) então, os primeiros **k** elementos do vetor serão inicializados com os respectivos valores e os restantes serão inicializados com o valor zero.

PT3: Crie um programa que inicializa com a seqüência {1,2,3,4,5} apenas 5 elementos de um **vetor** v de dimensão 10. Mostre todo o seu conteúdo após esta inicialização.

1

45

```
#include <stdio.h>
main()
{ int i;
  int v[10]={1,2,3,4,5};

  printf("v[i] = ");
  for (i=0; i< 10; i++)
  {
    printf("%d ",v[i]);
  }
}
```

Por último, um vetor poderia também ser declarado e inicializado com o comando:

```
int v[ ] = {5,10,15};
```

Este comando equivale a criar um vetor com apenas 3 espaços de memória, ou seja:

```
int v[3] = {5, 10, 15};
```

Ainda existe outra forma de inicializar os elementos de um vetor:

```
int v[6] = { 5 };
```

Neste caso, apenas o primeiro elemento será inicializado com o valor 5.

PE3: Faça um programa que verifique a afirmação anterior.

CONSTANTES

Ao se escrever um programa é interessante organizar o mesmo de forma que uma pequena alteração nas especificações iniciais não provoque grandes transformações no código. Em particular, imagine que no **PT2** ao invés de se usar um vetor de 5 elementos fosse necessário usar um vetor com 25. Uma solução trabalhosa seria substituir o valor 5 por 10. Para um programa pequeno esta é uma solução viável, mas não seria interessante para um programa com milhares de linhas de código. Além disso, esta abordagem pode ocasionar erros de substituição. Uma solução é definir um nome correspondente a um valor fixo, ou seja uma constante. A definição de uma constante é realizada após as linhas dos **#includes**, usando ou o comando **const** ou o comando **define**, como mostrado a seguir.

```
const tipo simbolo = valor;
```

```
#define simbolo valor
```

Quando a constante é declarada por **const** ele existe fisicamente em uma determinada posição de memória. Já quando o **define** é utilizado, o compilador irá substituir todas as ocorrências do símbolo pelo valor definido. As constantes definidas com o símbolo **#define** chamam-se constantes simbólicas.

PT4: Refaça o **PT2** utilizando **const** e **define** e observar as diferenças de sintaxe.

```
#include <stdio.h>
const int num = 10;
main()
{
  int i;
  int v[num];
  for (i=0; i< num; i++)
  {
    v[ i] = i + 1;
  }

  printf("v[ i ] = ");
  for (i=0; i< num; i++)
  {
    printf("%d",v[ i ]);
  }
}
```

```
#include <stdio.h>
#define num 10
main()
{
  int i;
  int v[num];
  for (i=0; i< num; i++)
  {
    v[i] = i + 1;
  }

  printf("v[i] = ");
  for (i=0; i< num; i++)
  {
    printf("%d",v[i]);
  }
}
```

PE4: Escrever um programa que declara um **vetor** com n= 10 números reais e coloque no i-ésimo elemento o resultado de $i^*(n-i)$.
Obs: O termo **i-ésimo** significa cada posição **i** do vetor.

2

46

PE5: Fazer um programa para ler um vetor A de dimensão 20 e a seguir calcular e imprimir o valor de S, onde:

$$S = (A_1 - A_{20})^2 + (A_2 - A_{19})^2 + ... + (A_{10} - A_{11})^2$$

PT5: O programa a seguir simula 60 lançamentos de um dado de 6 faces e armazena no vetor freqüência o número de vezes que uma face foi sorteada.
Obs: Para simplicidade, aqui se consideram que o zero é a primeira face, o 1 é a segunda face e assim por diante.

```
#include <time.h>
const int lanc = 60;
main()
{
 int frequencia[6] = { 0 };
 int i, face;
 // inicializa gerador aleatório.
 srand(time(0));
 for ( i = 0 ; i < lanc ; i++ )
 {
   // Fornece inteiro entre 0 e 5.
   face = rand()%6;
   freqencia[face]++;
 }
}
```

Observe que no **PT5** foi utilizado o recurso de gerar um número aleatório. A função **rand()** fornece um inteiro entre zero e **RAND_MAX** (uma constante definida na biblioteca <stdlib.h> e igual à 32767919). Para que o número retornado por esta função esteja dentro de um intervalo [0,b], basta usar o operador resto da divisão inteira (%) no seguinte comando: **rand()%(b+1)**. Para se obter um número no intervalo [a,b] basta fazer: a + (**rand()%(b+1)**). Observe, porém, que sem o comando **srand()** os números sorteados são sempre os mesmos.

PT6: Execute 3 vezes o programa a seguir e verifique que ele sempre fornece a mesma seqüência de números apesar de usar **rand()**:

```
#include <time.h>
const int lanc = 10;
main
{
 int i, face;
 for ( i = 0 ; i < lanc ; i++ )
 {
   face = rand()%6;
   printf("%d ",face);
 }
}
```

A função **rand()** parte sempre de uma mesma semente para realizar o cálculo dos números aleatórios, o que gera sempre seqüências aleatórias iguais. Para contornar este problema pode-se usar como semente um valor gerado pelo relógio do computador. Assim, foi utilizado o comando **time(0)** que fornece o número de segundos desde 01/01/1970 e está contido na biblioteca **<time.h>**.

Por fim, observe que no **PT5**, o primeiro elemento do vetor freqüência armazena o número de vezes que a face 1 foi sorteada, mas este valor fica na posição [0] do vetor:

face 1	face 2	face 3	face 4	face 5	face 6
10	11	9	9	11	10
v[0]	v[1]	v[2]	v[3]	v[4]	v[5]

PE6: Modifique o **PT5** para exibir os resultados contidos no vetor na forma de um histograma que indica o número de vezes que uma face foi sorteada.

Como visto no **PT6** e **PE6** vetores são particularmente importantes para agruparem dados e construir estatísticas sobre os mesmos. Isto pode ser mais bem observado com o **PT7** a seguir:

PT7: O programa a seguir lê o número de alunos de uma turma bem como as notas de cada aluno. Terminada a leitura de dados o programa encontra e imprime:
(1) A menor nota.
(2) A maior nota.
(3) A média aritmética: $\bar{x} = \sum_{i=1}^{n} x_i$

```
#include <stdlib.h>
#include <stdio.h>
main()
{
  int i, num;
  float menor, maior, media, notas[50];

  printf("Insira n. alunos:");
  scanf("%d",&num);
  for (i=0; i < num; i++)
  { // Leitura de num notas !
    printf("\n Insira %d nota: ",i+1);
    scanf("%f",&notas[i]);
```


47

```
  }
 // Encontrando menor valor.
  menor = notas[0];
  for(i=1; i < num; i++)
    if (notas[i] < menor)
      menor = notas[i];

 // Encontrando maior valor.
  maior = notas[0];
  for(i=1; i < num; i++)
    if (notas[i] > maior)
      maior = notas[i];

 // Encontrando valor médio.
  media = 0;
  for(i=0; i < num; i++)
    media = media + notas[i];
  media = media/num;

  // Impressao dos resultados.
  printf("\n\n-------Estatisticas--------\n");
  printf("Menor nota = %f \n",menor);
  printf("Maior nota = %f \n",maior);
  printf("Nota media= %f \n",media);
  system("PAUSE");
}
```

PE7: Aplicar os conceitos do **PE6** no **PT7** de forma a construir um histograma que verifica a quantidade de notas pertencentes a uma determinada faixa de valores. Para cada faixa de valores é associado um conceito tal como descrito Tabela **T1**:

TABELA T1

Faixa	Conceito
9 ≤ nota ≤ 10	A
7 ≤ nota < 9	B
5 ≤ nota < 7	C
3 ≤ nota < 5	D
0 ≤ nota < 3	E

Depois de totalizar o número de valores pertencentes a cada faixa, o programa deverá mostrar com um histograma o número de notas pertencentes a cada conceito.

Exemplo de execução:
Insira n. de alunos: 5
Insira nota 1: 10
Insira nota 2: 8
Insira nota 3: 7
Insira nota 4: 6
Insira nota 5: 8
Histograma:
A: *
B: ***
C: *
D:
E:

PROGRAMAS BÁSICOS

PB1: Construa um programa que coloque em um vetor os 10 primeiros valores da seqüência de Fibonacci. Dica: a seqüência de Fibonacci é definida pela seguinte fórmula recursiva:

$$f(n) = \begin{cases} 1, \text{se } n = 0 \\ 1, \text{se } n = 1 \\ f(n-1) + f(n-2), \text{c.c.} \end{cases}$$

PB2: Altere o **PT5** para permitir 1000 lançamentos de um dado de 10 faces.

PB3: Fazer um programa para ler um vetor V de elementos inteiros e de dimensão M. Verificar se a soma dos elementos pares (posição par) do vetor é igual a soma dos elementos ímpares do vetor, imprimindo mensagens adequadas. Ler dois valores genéricos X e Y e verificar quantas vezes esses valores aparecem juntos, e nesta ordem, no vetor V. Imprimir as respostas.

PB4: Fazer um programa para ler um vetor V de elementos inteiros e de dimensão N. Separar esse vetor V em dois vetores A e B. O vetor A deve conter os elementos positivos de V e o vetor B os elementos negativos de V. Os elementos nulos de V não devem ser gravados, mas devem ser contados. Escrever os vetores V, A e B. Escrever quantos elementos nulos foram encontrados em V.

PB5: Fazer um programa para ler um vetor A e sua dimensão N e a seguir:
a) Calcular e imprimir o valor de S, sendo

$$S = \sum_{i=1}^{N} \frac{i}{a_i}$$

onde a_i é o i-ésimo valor armazenado na variável A
b) Calcular e imprimir quantos termos da série têm o numerador inferior ao denominador

48

OBS: Matematicamente a somatória começa com $i=1$ e vai até $i=N$. Computacionalmente o vetor A começa com o índice zero. Logo tem que ser feita uma adaptação na fórmula para que o primeiro elemento da série não fique igual a zero, isto é, no denominador tem que se usar $(i+1)$. Não se esqueça de transformar o $(i+1)$ em float para que a divisão seja feita corretamente.

PB6: Fazer um programa para ler dois vetores X e Y, ambos de dimensão N e ambos compostos de elementos inteiros. Calcular o valor de T dado da seguinte maneira:

$$T = \frac{X[n]}{Y[1]} - \frac{X[n-1]}{Y[2]} + \frac{X[n-2]}{Y[3]} - \ldots + \frac{X[1]}{Y[n]}$$

Imprimir os vetores X e Y e o valor de T.

PB7: Seja o polinômio P dado pela formula a seguir:

$$P = a_1 x^{n-1} + a_2 x^{n-2} + a_3 x^{n-3} + \ldots + a_{n-1} x + a_n$$

Fazer um programa para ler o valor de N e ler os coeficientes, $a_i, i = 1,2,\ldots N$ do polinômio e a seguir calcular o valor de P para 10 valores de x lidos, de modo que P[1] seja o valor para o 1°. valor de x, P[2] seja o valor para o 2°. valor de x e assim por diante. Imprimir para cada valor de x o valor de P correspondente.

PB8: O Bubble sort é um algoritmo que ordena em ordem crescente os elementos de um vetor. Para tanto, os valores menores "sobem" gradualmente para o topo do vetor, da mesma forma que bolhas de ar sobem na água, enquanto valores maiores afundam (submergem) para a parte de baixo do vetor. O Bubble sort varre (n-1) vezes todo o vetor, comparando os elementos dois a dois, (n-1) vezes. A cada varredura, se um par esta em ordem crescente, nada é feito. Caso contrário, os elementos no vetor são permutados. Implementar o Bubble sort. Um exemplo de execução é dado por:

Vetor inicial: 8 5 1
Varredura 1:
Comparacao 1: 8 5 1 → 5 8 1
Comparação 2: 5 8 1 → 5 1 8

Varredura 2:
Comparacao 1: 5 1 8 → 1 5 8
Comparação 2: 1 5 8 → 1 5 8

PB9: Construa um programa que coloque em um vetor os 10 primeiros valores das sucessivas aproximações do valor de π utilizando a seguinte fórmula:

$$\pi = 4\left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + \ldots\right)$$

Assim, o elemento v[0] deve conter apenas o primeiro termo da seqüência que aproxima o valor de π, ou seja, o valor 4. O elemento v[1] deve conter o resultado de 4 - 4/3 e assim por diante até o décimo elemento.

PB10: O resultado k do produto interno de dois vetores v e t de dimensões 1 × n é calculado através da seguinte fórmula:

$$k = < v,t > = \sum_{i=1}^{n} v_i t_i$$

Escreva um trecho de código, que calcula o produto interno de dois vetores com n elementos.

PB11: Construa um programa que coloque em um vetor os n primeiros valores da soma de uma progressão geométrica cuja fórmula é dada por: $S_n = n(a_1 + a_n)/2$. O programa deve solicitar o valor de n, a_1 e q para o usuário.

PB12: Construa um programa que coloque em um vetor os n primeiros valores da soma de uma progressão geométrica cuja fórmula é dada por: $S_n = \frac{a_1(q^{n-1}-1)}{q-1}$. O programa deve solicitar os valores de n, a_1 e q para o usuário.

PB13: Use um vetor para resolver o seguinte problema. Uma empresa paga seis vendedores com base em comissões. O vendedor recebe um valor fixo de R$ 500,00 por mês mais 10 por cento de suas vendas brutas daquele mês. Por exemplo, um vendedor que teve vendas brutas de R$ 3000,00 em um mês recebe R$500,00 mais 10 por cento de R$ 3000,00, ou seja, um total de R$ 800,00. Escreva um programa (usando um vetor de contadores) que determine quantos vendedores receberam salários nos seguintes[5]

49

intervalos de valores (considere que o salário de cada vendedor é truncado para que seja obtido um valor inteiro):

Faixa 1	Faixa 2	Faixa 3	Faixa 4	Faixa 5
500 -999	1000 -1499	1500 -1999	2000 -2999	3000 - em diante

PB14: Refazer o **Exercício PB13**, mas considerando que para cada Faixa de valor de vendas existe um percentual de comissão como dado na seguinte Tabela.

	Faixa 1	Faixa 2	Faixa 3	Faixa 4	Faixa 5
Comissão	10%	12%	14%	15%	20%

PB15: Elaborar um sistema de reservas de companhias áreas em que os assentos de cada vôo são representados por um vetor. Inicialmente todos os assentos estão vagos e todos os elementos do vetor possuem valor 0. A cada reserva realizada a posição do vetor correspondente ao assento deverá apresentar o valor 1. Assim, para cada usuário que utiliza o sistema deverá ser apresentado o seguinte menu:

Favor digitar:
1 – Para verificar ocupação do avião.
2 – Realizar reserva de um assento.
3 – Cancelar reserva de um assento.

Caso o usuário digite a opção 1, deverá ser mostrada na tela quais são as posições do avião que estão disponíveis (lembre-se que a posição v[0] do vetor corresponde ao assento número 1 do avião e assim por diante). Caso a opção 2 seja selecionada, então, deverá ser verificado se a posição está ocupada ou não. Se estiver, imprimir uma mensagem falando isso. Caso contrário, realizar a reserva e mostrar a nova ocupação do avião. Por último, caso a opção 3 seja selecionada; então, verificar se o assento está ocupado. Se estiver, então, tornar o assento vazio e mostrar a nova ocupação do avião. Caso contrário, imprimir uma mensagem de que o assento já está disponível.

6

50