COMPUTAÇÃO
UFBA

Universidade Federal
da Bahia

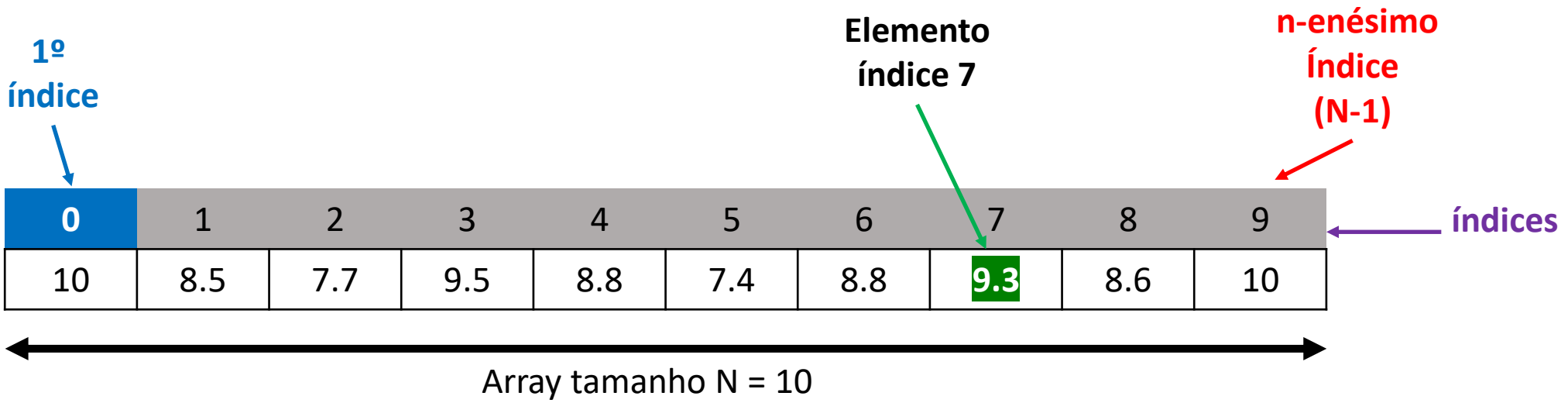
Laboratório de Programação I

Prof.: Claudio Junior N. da Silva (claudiojns@ufba.br)

Vetores

Vetores

- Conjunto de elementos consecutivos, do mesmo tipo, que podem ser acessados individualmente a partir de um único nome




```
float notas[10];
notas[0] = 10;
notas[7] = 9.3;
```

Vetores - Declaração

- tipo nome_variavel [n_elementos];
- Onde:
  - tipo = Tipo de dados de cada elemento do vetor;
  - Nome_variável = Nome pelo qual o vetor será conhecido;
  - n_elementos = Quantos elementos tem o vetor.
- Antes:
  - float nota1, nota2, nota3, ... , nota100;
- Depois
  - **float notas[100];**

Vetores - Declaração

- Um vetor pode conter elementos de qualquer tipo de dados. No entanto, os elementos do vetor são todos do mesmo tipo, o qual é definido na declaração do mesmo
- Ex:
  - `int g[20]; /*g é um vetor com 20 números inteiros*/`
  - `float renda[100];`
  - `Ou ainda:`
    - ```
      N = 5; tam = 30;
      int idade[N];
      char str[20*tam];
      ```
      ```

Vetores – Índices – cont.

- Pode ser representado por qualquer expressão que devolva um valor inteiro
- Ex:
  - vetor [1] = 14;
  - vetor [3-1] = 10;
  - vetor [1 + vetor [1]] = 513;

Vetores – Carga de Valores

- Quando são criados contém valores aleatórios (lixo) em cada uma das suas posições
- É possível iniciar automaticamente todos os elementos de um vetor
- Ex:
  - `tipo var[n] = {valor1, valor2, valor3...};`
  - `int v[3] = {0, 1, 2};`
- Se um vetor for declarado com n elementos e forem colocados apenas k valores (k <n), os primeiros k elementos do vetor serão iniciados com os respectivos valores e os restantes com o valor zero
- Ex:
  - `int v[10] = {10, 20, 30};`

```
/* equivale a int v [10] = {10, 20, 30, 0, 0, 0, 0, 0, 0, 0} */
```

Vetores – Carga de Valores

- A declaração e a carga inicial de um vetor podem ser realizadas sem indicar qual o número de elementos do vetor. Neste caso, o compilador vai criar um vetor com tantos elementos quantas as cargas iniciais;
- Não se pode declarar vetores sem dimensão:
  - Ex:
    - `tipo var[] = {valor1, valor2, valor3...};`
  - No entanto:
    - `int v[] = {10, 20, 30}; /*está correto*/`
    - `int v[]; /*está incorreto e provoca erro compilação*/`
- Para percorrer um vetor:
  - `for (i=0; i < N; i++)  // sabemos a quantidade/valor de N`
  - `while (condição){...}  // não sabemos o valor de N`

Vetores – Praticando

- Um pesquisador solicitou a você, desenvolvedor do centro de pesquisas, para criar um programa que pudesse ler e imprimir a idade de 10 pessoas;
- Sua tarefa é desenvolver esse programa. Vamos lá....

```
#include <iostream>
using namespace std;

int main () {
    int i, vet[10];
    cout << ”Informe a idade das pessoas: ” << endl;
    for (i=0; i < 10; i++) //vai de 0 a 9
        cin >> vet[i];

    for (i=0; i < 10; i++) //vai de 0 a 9
        cout << ”Idade da pessoa " << i << ": " << vet[i] << endl;
}
```

Vetores – Praticando

- Seu programa está funcionou muito bem e atendeu os requisitos do pesquisador. Mas agora a pesquisa evoluiu e o pesquisador não tem como determinar a quantidade de pessoas que serão pesquisadas;
- Ele te procurou e você terá de alterar o programa, ou seja, seu programa deverá continuar informando a idade das pessoas até que o processo seja interrompido;
- Você decidiu não a estrutura de repetição anterior (for), agora seu programa irá utilizar outra estrutura (while? doc... while?) . Vamos lá????

Vetores – Praticando

```
#include <iostream>
using namespace std;

int main () {
   int i=0, vet[100], cont=0, ler=1;

   while (ler){
      cout << " Informe a idade da pessoa ou digite 0 para sair: ";
      cin >> ler;
      if (!ler)
         break;
      vet[i] = ler;
      cont++;
      i++;
   }

 for (i=0; i < cont; i++)

      cout << ”Idade da pessoa " << i << ": " << vet[i] << endl;

}
```

Classe Vector

Vector

- A classe vector é uma alternativa à representação de array primitivo;
- Template de classe:
  - Necessário especificar o tipo dos elementos;
  - vector<int> vx.
- Necessário incluir o ficheiro “vector.h”
  - #include <vector>

Vector

- Definição de um vector com/sem tamanho determinado:

```
int main() {
    const int tam=10;
    vector<int> v1(tam);
    vector<int> v2;    // vazio
    int v2[tam];
    ...
    for (int i=0; i<tam; i++)
         v2[i]=v1[i]
}
```

- Elementos são iniciados com valor de defeito do tipo. Pode-se atribuir um valor inicial:

```
vector<int> v1(10,-1);      // v1 contém 10 elementos do
                            // tipo inteiro inicializados a -1
```

Vector - Métodos

Exemplos de Métodos	Descrição
vx.size();	Retorna tamanho do vector
vx.empty();	Determina se vector está vazio
vx.resize(novo_tamanho);	Redimensiona vector
vx2 = vx;	Cópia de vectores
vx.clear()	Remove todos os elementos de um vector
vx.erase()	Remove elementos de um vector
vx.insert()	Insere elementos em um vector
vx.max_size()	Retorna o número máximo de elementos que o vector pode conter
vx.reserve()	Define o número mínimo de elementos que o vector pode conter
vx.resize()	Altera o tamanho do vector
vx.capacity()	Retorna o número de elementos que o vector pode conter

Vector

- Inserir um elemento:

```
vy.push_back(x);    // insere x no fim do vector
vy.pop_back(x2);    // retira x2 do fim do vector
```

- Uso de iterador:

```
iterator it;          // declaração
it = vy.begin();      // aponta para o 1º elemento
it = vy.end();        // aponta para o último elemento
*it                   // elemento do vector referenciado pelo iterador
it++                  // incrementa iterador, aponta para o próximo elemento
```

Entendo *push back, size e clear*

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> v;
    int x;

    cout << "Informe os elementos do Vector ou -1 para encerrar:" << endl;
    while(cin >> x && x >= 0)
        v.push_back(x);   // função push_back adiciona elemento no final e aloca espaço

    cout << "Os elementos do Vector são:" << endl;
    for(int i=0; i < v.size(); i++) // size retorna a quantidade de elementos
       cout << v[i] << endl;

    v.clear();                    // função clear remove todos os elementos de um vector

    cout << "Informe novos elementos do Vector ou -1 para encerrar:" << endl;
    while(cin >> x && x >= 0)
        v.push_back(x);

    cout << "Os elementos do Vector agora são:" << endl;
    for(int i=0; i < v.size(); i++)
        cout << v[i] << endl;
}
```

Entendo *erase*

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    vector<int> v;
    int x;

    cout << "Informe os valores do vector ou 0 para finalizar: " << endl;
    while(cin >> x && x > 0)
        v.push_back(x);

    cout << "Os valores informados foram : " << endl;
    for(int i=0; i < v.size(); i++) // size retorna a quantidade de elementos
       cout << v[i] << endl;        // o acesso é realizado normalmente

    v.erase(v.begin());             // apaga o primeiro elemento
    v.erase(v.begin()+1);           // apaga o segundo elemento
    v.erase(v.begin()+2, v.end()); // apaga todos os elementos exceto os dois primeiros

    cout << "Após a exclusão, os valores são : " << endl;
    for(int i=0; i < v.size(); i++) // size retorna a quantidade de elementos
       cout << v[i] << endl;        // o acesso é realizado normalmente
}
```

Vector – Ordenação - sort

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // sort -> std

int main() {
    vector<int> v;
    cout << "Informe os dados do vector: " << endl;
    for(int i=0; i < 5; i++) {
        int j;
        cin >> j;
        v.push_back(j);
    }
    sort(v.begin(), v.end()); // ordenação com complexidade ~ NlogN (não estável)

    // vector ordenado
    cout << "O vector ordenado é: " << endl;
    for(int i=0; i < v.size(); i++){
        cout << v[i] << endl;
    }
}
```

Vector – Ordenação – stable_sort

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std; // sort -> std

int main() {
    vector<int> v;
    cout << "Informe os dados do Vector: " << endl;
    for(int i=0; i < 5; i++) {
        int j;
        cin >> j;
        v.push_back(j);
    }
    stable_sort (v.begin(), v.end()); // ordenação com complexidade = NlogN

    cout << "Vector ordenado: " << endl;
    for(int i=0; i < v.size(); i++){
        cout << v[i] << endl;
    }

}
```

Tarefa

- Depois do sucesso do seu programa para ler e imprimir as idades das pessoas pesquisadas, o pesquisador te procurou pedindo informando que seu programa agora precisa:

1. Ler um número indeterminado de idade das pessoas, ou seja, ele quer continuar registrando entrada dos dados até digitar, por exemplo, uma idade < 1;
2. Seu programa deve imprimir uma relação de todos os dados registrados, indicando a ordem e a pessoa e a idade;
3. Seu programa deve calcular e depois imprimir a média das idades, a maior idade e a menor idade;
4. Seu programa deve informar quantas pessoas tem idade maior ou igual à média das idades e quantas pessoas tem idade menor do que a média das idades;
5. Seu programa deve imprimir a relação das idades em ordem crescente e em ordem decrescente.

Vector – Exemplo Iterator

```
#include <iostream>
#include <vector>
using namespace std;

int main() {
    const int tam=7;
    vector<int> v1;
    int v2[tam] = {0,1,1,2,3,5,8}; // Inicialização de array primitivo
    for (int i=0; i < tam; i++)
          v1.push_back(v2[i]);

    cout << "Conteúdo do vector v1 : \n";
    for (vector<int>::iterator it = v1.begin(); it != v1.end();  it++){
        cout << *it << " "; // valor na posição apontada por it
    }
    cout << endl;

    cout << "Conteúdo do vector v1 : \n";
    for (int i = 0; i < sizeof v2/sizeof v2[0]; i++){
        cout << v2[i] << " ";
    }
}
```