Vectores

Mestrado Integrado em Engenharia Informática FCT UNL
http://ctp.di.fct.unl.pt/miei/ip/

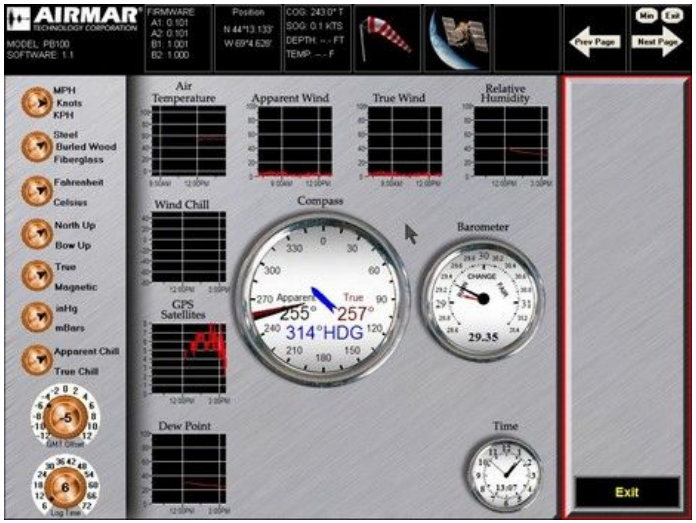
Corpo Docente 2020/2021
António Ravara, Artur Miguel Dias, Bernardo Toninho,
Ema Vieira, Inês Fernandes, Margarida Mamede,
Miguel Monteiro, Rui Nóbrega

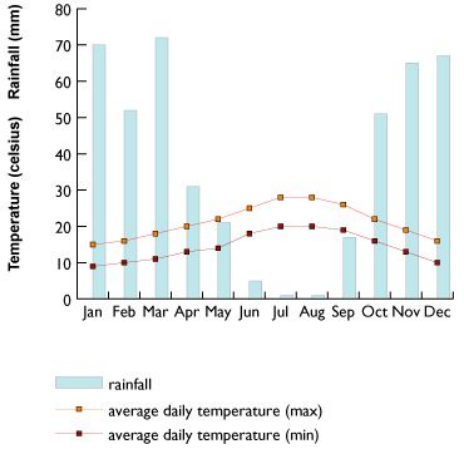
Vectores

- Neste capítulo, vamos programar classes que necessitam **armazenar** e **processar** diversos **dados de um mesmo tipo**.
- Pelo caminho, será introduzido um tipo de dados estruturado: **vector**
- Em relação ao **vector** vamos aprender:
  - como se **declaram** e **inicializam**
  - quais as suas principais **limitações**
  - como se **acede** aos **dados** neles organizados

Vectores

- Sendo um tipo de dados que serve para organizar e estruturar a informação, vamos aprender também a:
  - inserir um elemento
  - remover um elemento
  - pesquisar/procurar um elemento
  - ordenar os elementos





Estação Meteorológica com Memória

E se quisermos consultar as diferentes amostras de temperaturas registadas?

Temos de armazenar as amostras de temperatura registadas?

Estação Meteorológica com Memória

- Cada objecto `WeatherStationMem` recebe valores de temperatura ao longo do tempo e guarda cada uma das amostras.
- Operações reconhecidas

```
void sampleTemperature(double temp)
```
Registar a amostra `temp` na estação

```
double getMaximum()
```
Consultar a máxima temperatura observada até ao momento

pre: `numberTemperatures()> 0`

```
double getMinimum()
```
Consultar a mínima temperatura observada até ao momento

pre: `numberTemperatures()> 0`

```
double getAverage()
```
Consultar a média das temperaturas observadas até ao momento

pre: `numberTemperatures()> 0`

```
double getSample(int i)
```
Consultar a i-ésima amostra registada

pre: `i > 0 && i <= numberTemperatures()`

```
int numberTemperatures()
```
Consultar o número total de amostras registadas.

Exige que a estação contenha todas as temperaturas antes registadas por meio de `sampleTemperature`

Estação Meteorológica com Memória

- Interacção com o utilizador:
  - O programa deve receber várias medições de temperatura
  - O programa deve terminar quando receber um valor não numérico
  - O programa deve listar, no final, todas as amostras registadas

Estação Meteorológica com Memória

O método `sampleTemperature()` tem de armazenar todas as amostras de temperatura.

Mas onde podemos guardar as diferentes amostras que vamos obtendo?

- Em diversas variáveis de instância que declaramos para o efeito?

```
private double sample1, sample2, sample3, ... ?
```

- Mas como sabemos quantas variáveis declarar?

  Fixamos um valor máximo à partida?

- E quando queremos guardar uma amostra, como decidimos em que variável deverá o valor ser guardado?

```
if (count == 1) sample1 = temp;
  else if(count == 2) sample2 = temp;
       else if(count == 3) sample3 = temp;
...
```

?

Estação Meteorológica com Memória

- Mas, então, onde podemos guardar as diferentes amostras que vamos obtendo?

Estação Meteorológica com Memória

- Mas, então, onde podemos guardar as diferentes amostras que vamos obtendo?
  - Em vectores (*arrays*)

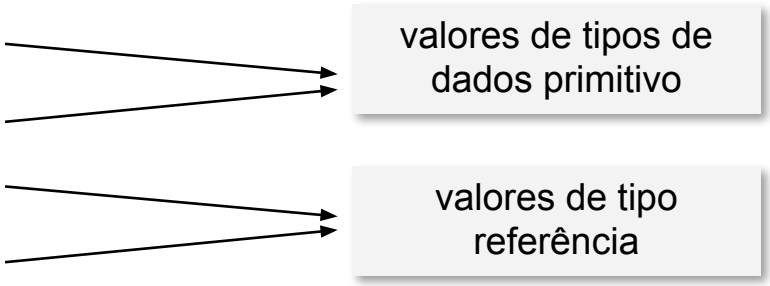
Estação Meteorológica com Memória

- Mas, então, onde podemos guardar as diferentes amostras que vamos obtendo?
  - Em vectores (*arrays*)
- O que é um *vector* em programação?

Estação Meteorológica com Memória

- Mas, então, onde podemos guardar as diferentes amostras que vamos obtendo?
  - Em vectores (*arrays*)
- O que é um *vector* em programação?
  - é uma entidade que serve para guardar e dar uma estrutura a uma série de dados de um mesmo tipo

    Exemplos:
    - vectores de valores **`int`**
    - vectores de valores **`double`**
    - vectores de objectos `Rect`
    - vectores de objectos `Circle`

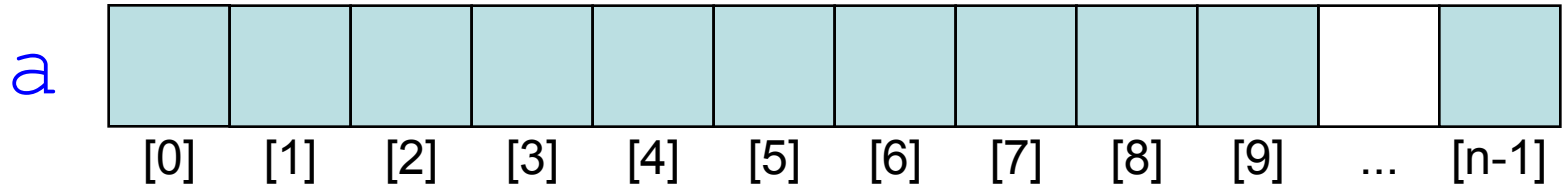


- Tem uma dimensão fixa

Vectores

- Mas que estrutura é essa?
  - Os elementos são identificados por intermédio de um índice!

    a = (a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, …, a_{n-1})
  - *n* é a dimensão do vector
  - Em matemática, a_i é o elemento do vector *a* com índice *i*.
  - Em programação em Java, o elemento do vector *a* com índice *i* é identificado por `a[i]`. Note-se que 0 <= *i* < *n*



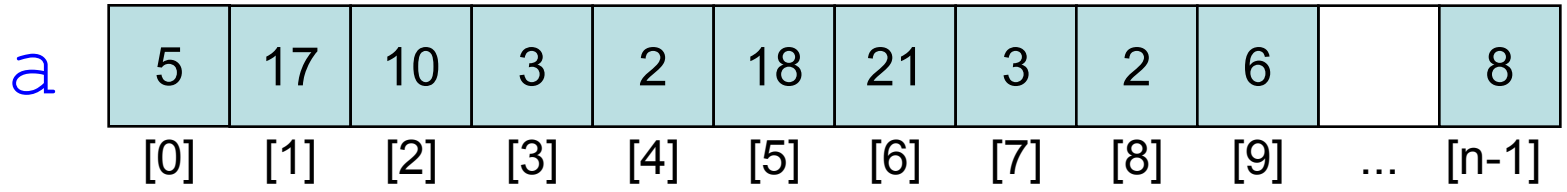
Vectores

- Como se guarda um valor num vector?
  - Para guardar um valor no vector precisamos em primeiro lugar saber em que posição o queremos guardar (índice)
  - Para escrever na posição de índice ***i*** do vector ***a*** bastará escrever:

```
a[i] = expr;
```

expr é uma expressão do mesmo tipo dos elementos do vector

  - Exemplo para um vector de valores de tipo `int`:



Vectores

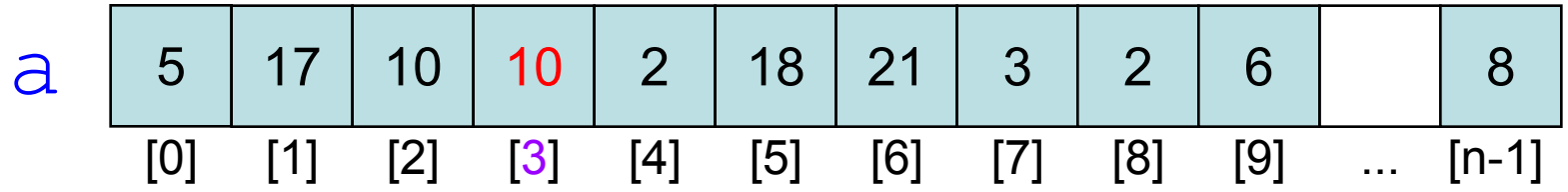
- Como se guarda um valor num vector?
  - Para guardar um valor no vector precisamos em primeiro lugar saber em que posição o queremos guardar (índice)
  - Para escrever na posição de índice ***i*** do vector ***a*** bastará escrever:

```
a[i] = expr;
```

expr é uma expressão do mesmo tipo dos elementos do vector

  - Exemplo para um vector de valores de tipo `int`:

```
a[3] = 10;
```



Vectores

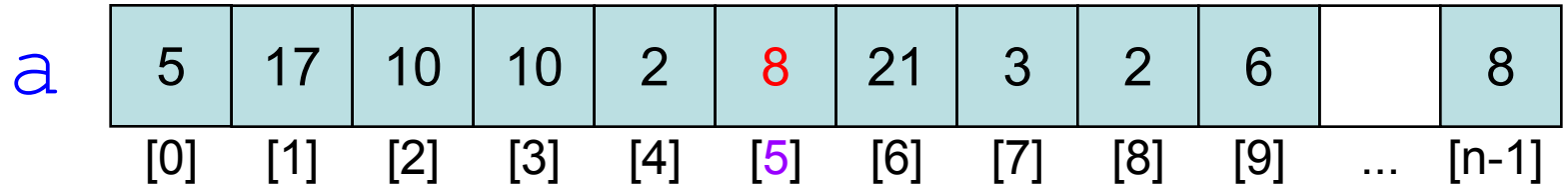
- Como se guarda um valor num vector?
  - Para guardar um valor no vector precisamos em primeiro lugar saber em que posição o queremos guardar (índice)
  - Para escrever na posição de índice ***i*** do vector ***a*** bastará escrever:

```
a[i] = expr;
```

expr é uma expressão do mesmo tipo dos elementos do vector

  - Exemplo para um vector de valores de tipo `int`:

```
a[3] = 10; a[5] = 8;
```



Vectores

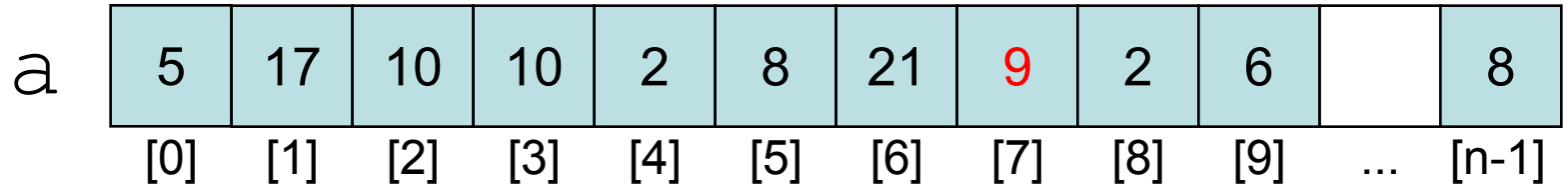
- Como se guarda um valor num vector?
  - Para guardar um valor no vector precisamos em primeiro lugar saber em que posição o queremos guardar (índice)
  - Para escrever na posição de índice ***i*** do vector ***a*** bastará escrever:

```
a[i] = expr;
```

expr é uma expressão do mesmo tipo dos elementos do vector

  - Exemplo para um vector de valores de tipo `int`:

```
a[3] = 10; a[5] = 8; a[7]=9;
```



Vectores

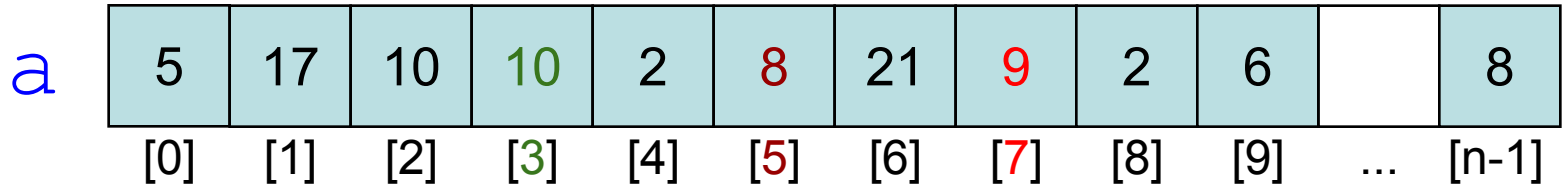
- Como se guarda um valor num vector?
  - Para guardar um valor no vector precisamos em primeiro lugar saber em que posição o queremos guardar (índice)
  - Para escrever na posição de índice **`i`** do vector **`a`** bastará escrever:

```
a[i] = expr;
```

expr é uma expressão do mesmo tipo dos elementos do vector

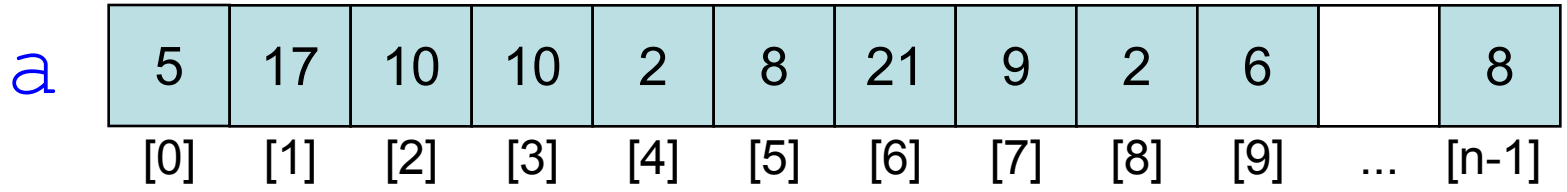
  - Exemplo para um vector de valores de tipo **`int`:**

```
a[3] = 10;        a[2+3] = 8;        se k=7, a[k]=9;
```



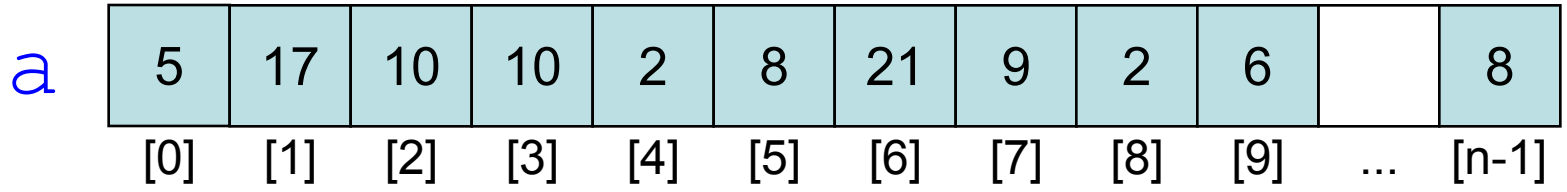
Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[*i*] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:

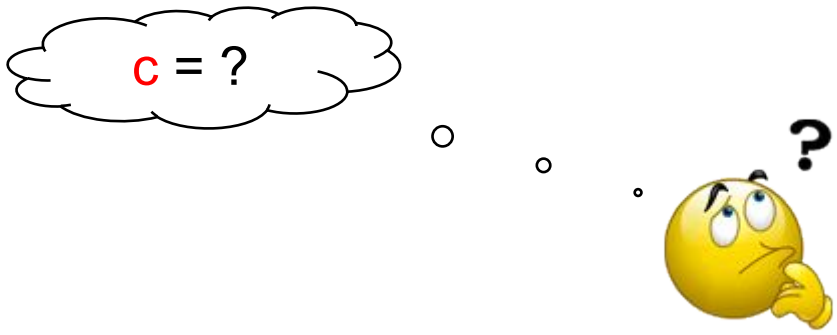


Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[*i*] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:

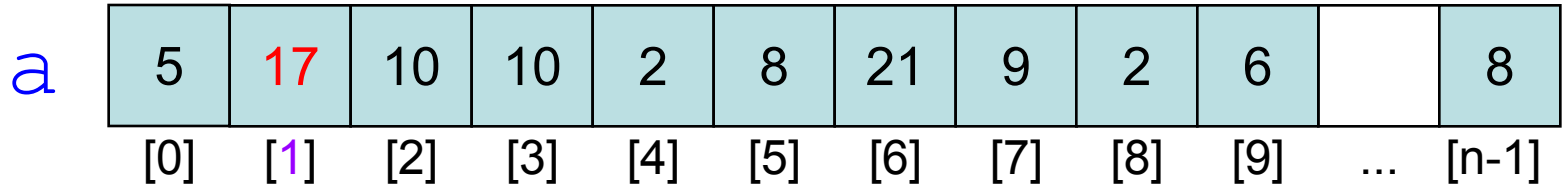



```
c = a[1];
```

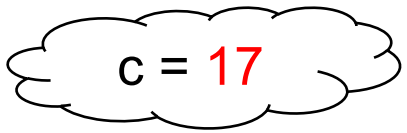


Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[*i*] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:

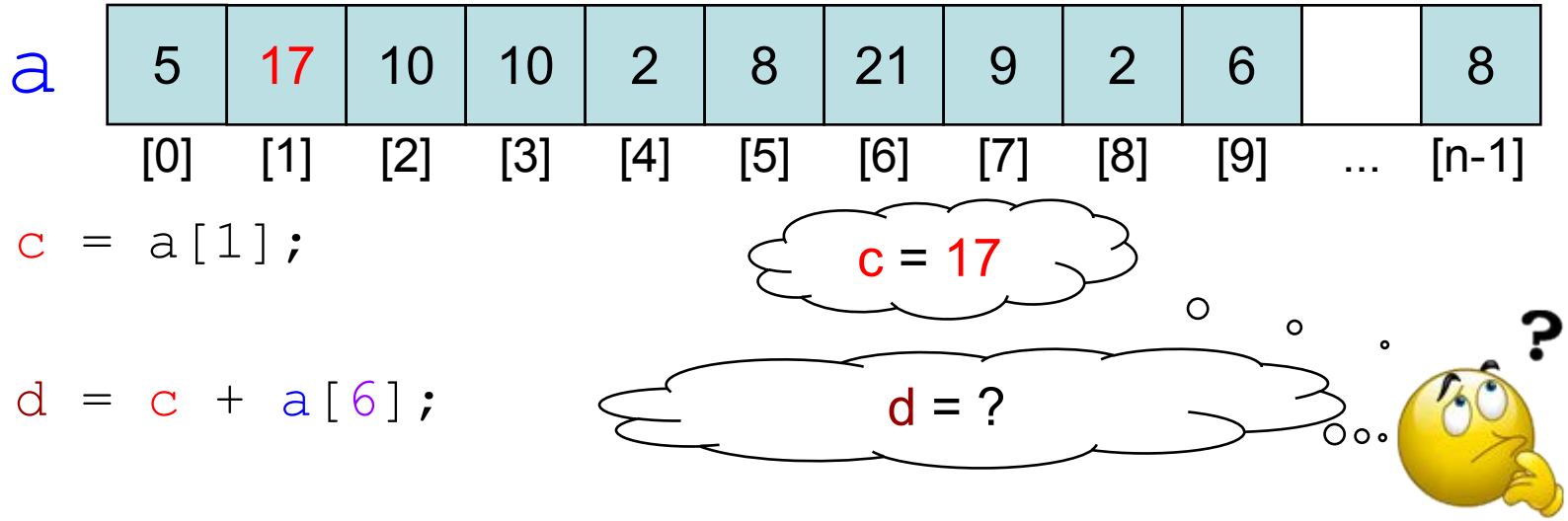



```
c = a[1];
```



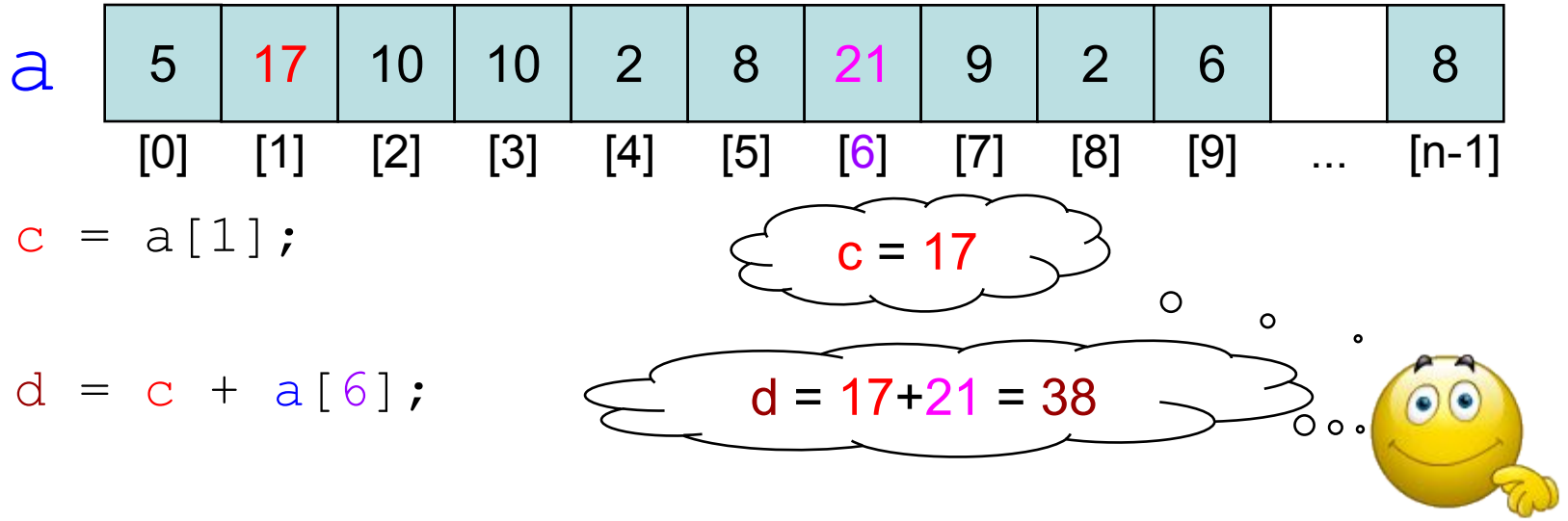
Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[***i***] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:



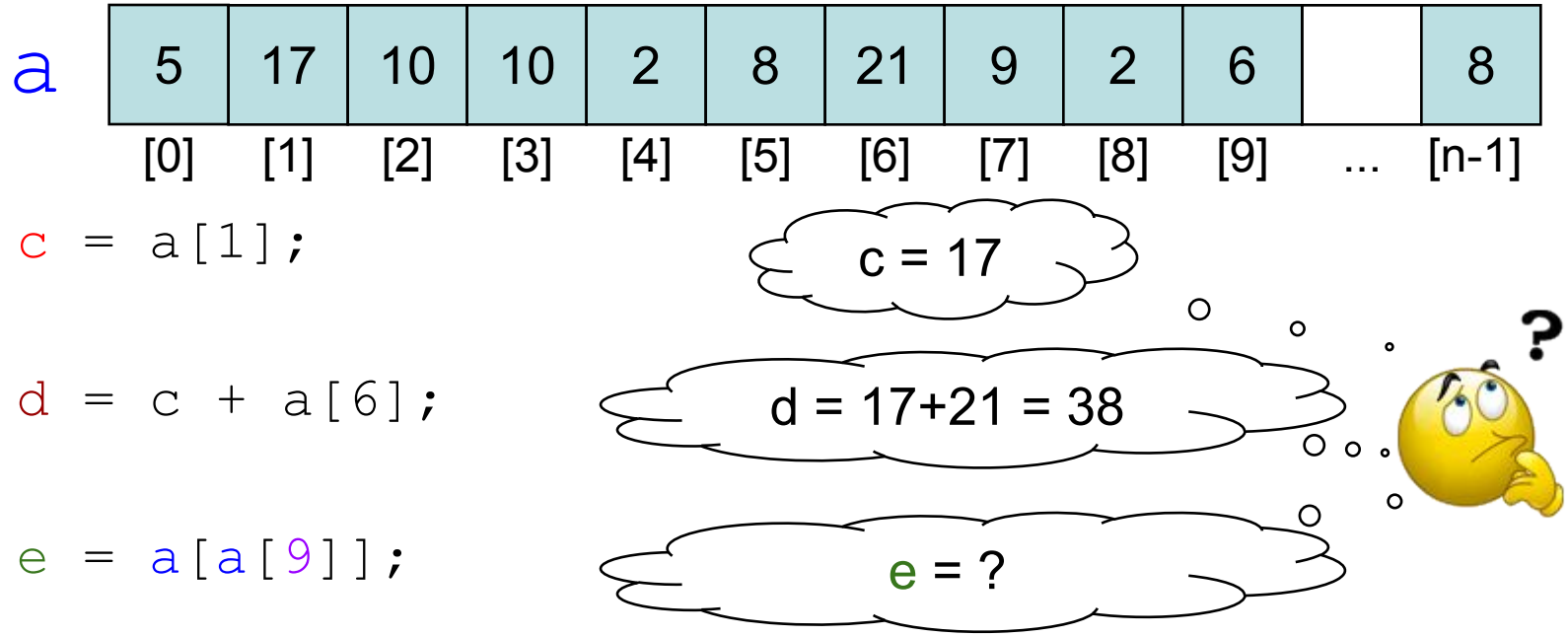
Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[***i***] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:



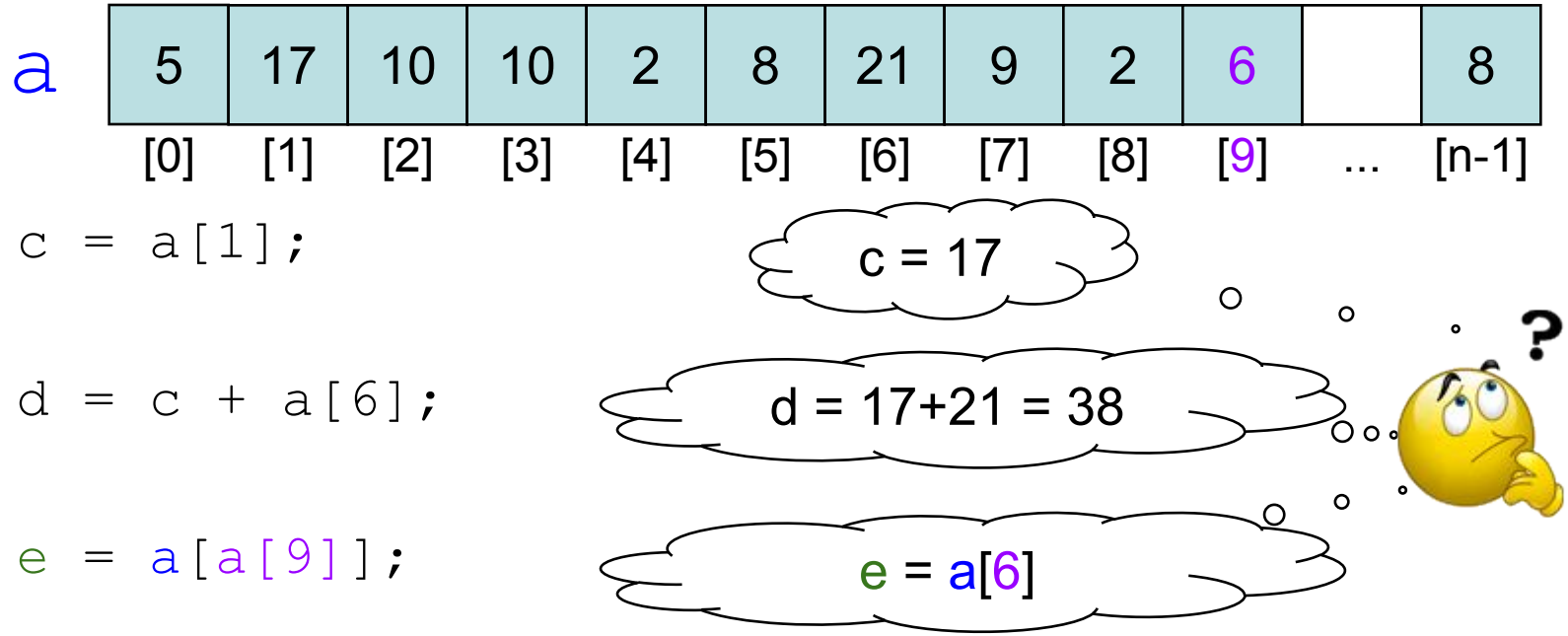
Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[***i***] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:



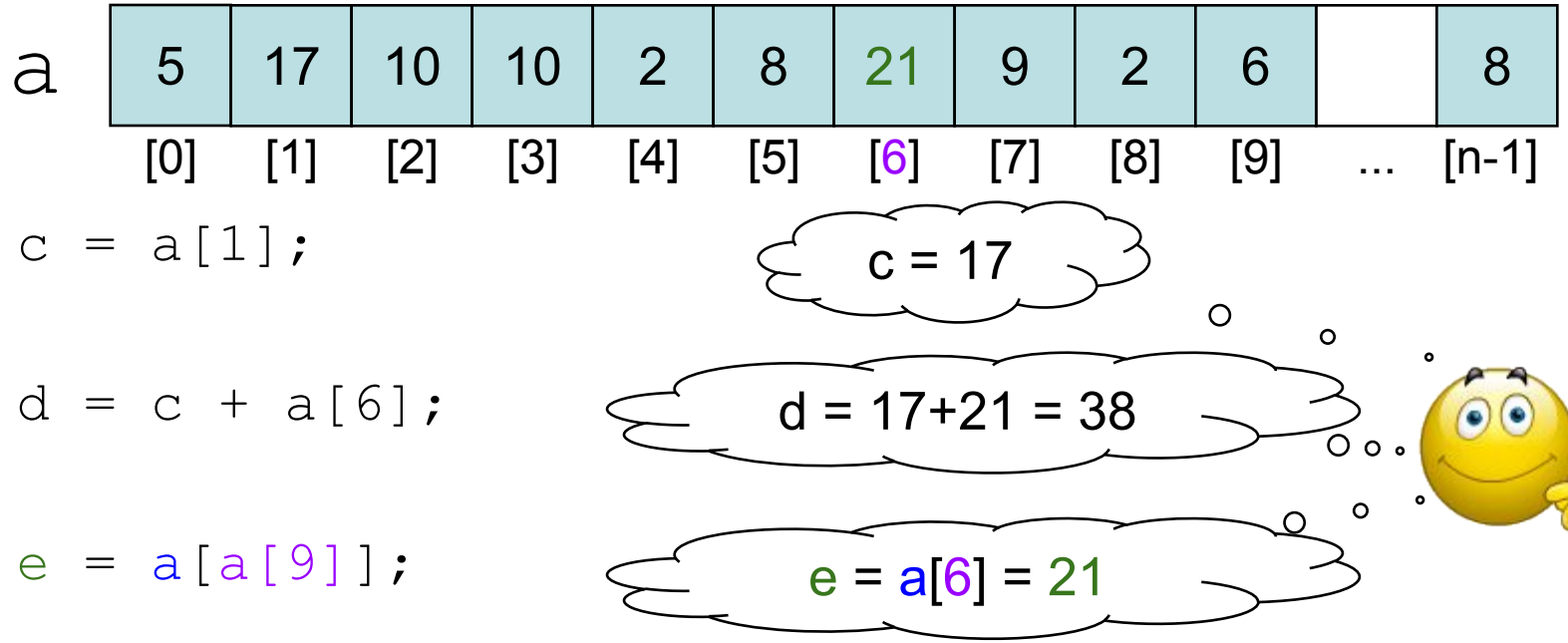
Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[***i***] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:



Vectores

- Como se consulta um valor guardado num vector?
  - Para consultar o elemento do vector **a** com índice **i** bastará escrever **a**[***i***] em qualquer expressão do tipo dos elementos do vector.
    - Exemplo para um vector de valores de tipo **int**:



Vectores

- Um vector é um espaço contíguo de memória
- O seu nome contém o endereço inicial
- O **índice do vector pode ser calculado por** qualquer expressão que seja avaliada para um inteiro entre `0` e `n-1`, sendo `n` a dimensão do vector
- Repare que `a[i]` se avalia como expressão do tipo dos elementos do vector `a`
- Note que a célula de memória correspondente a `a[i]` "é" `a + i*tam_tipo`
- Como `a` contém a primeira posição, o primeiro índice é 0

Vectores

- Como se declaram variáveis de tipo vector?

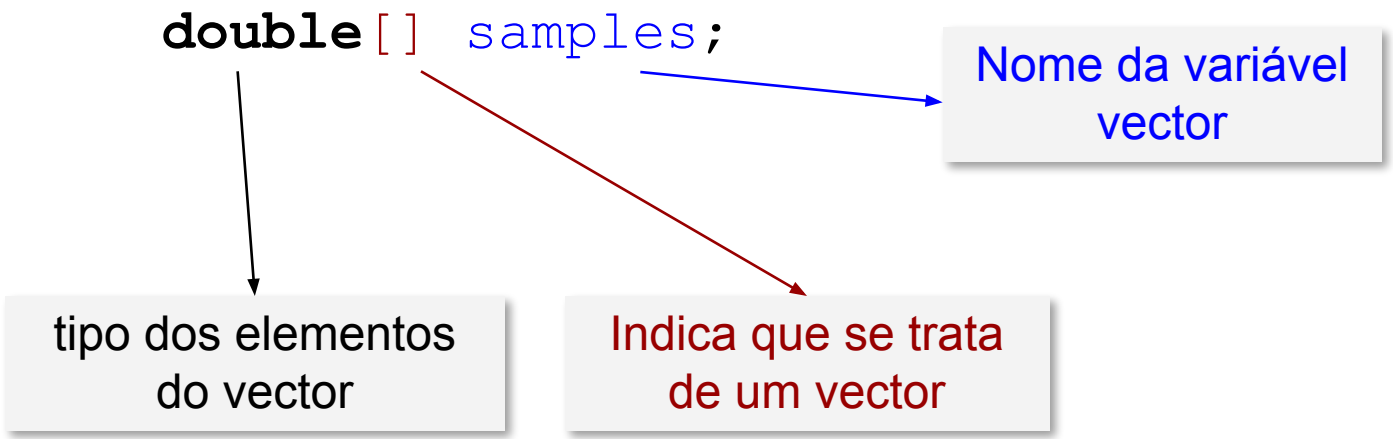
Vectores

- Como se declaram variáveis de tipo vector?
  - Exemplo de um vector para guardar valores de tipo **`double`**:

```
double[] samples;
```

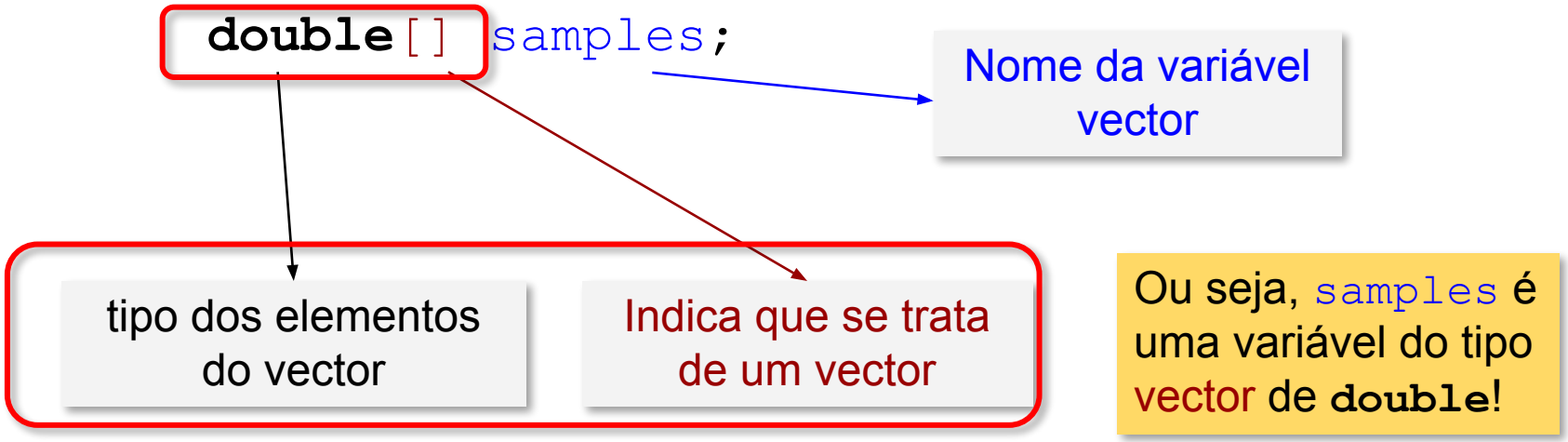
Vectores

- Como se declaram variáveis de tipo vector?
  - Exemplo de um vector para guardar valores de tipo `double`:



Vectores

- Como se declaram variáveis de tipo vector?
  - Exemplo de um vector para guardar valores de tipo `double`:



- Mas atenção, ainda falta criar o objecto vector:
  - `samples = new double[N];` → A constante `N` indica a capacidade do vector (número máximo de elementos que lá se podem guardar)

Estação Meteorológica com Memória
(Declaração do vector)

```
public class WeatherStationMem {

   // variaveis de instancia
   private int count;
   private double[] samples;

   // Construtores
   ...

   // Operacoes/Metodos
   ...
}
```

Vector que vai guardar as amostras de temperaturas registadas

Estação Meteorológica com Memória
(Inicialização do vector)

```
public class WeatherStationMem {
    private static final int DEFAULT_CAPACITY = 100;

    private int count;
    private double[] samples;
    // Construtores
    public WeatherStationMem() {
        count = 0;
        samples = new double[DEFAULT_CAPACITY];
    }
    // Operações/Métodos
    ...
}
```

Capacidade, por omissão, do vector de amostras

Criação inicial do vector `samples`, com dimensão ***DEFAULT_CAPACITY***.

Estação Meteorológica com Memória
(Inicialização do vector)

```
public class WeatherStationMem {
    private static final int DEFAULT_CAPACITY = 100;

    private int count;
    private double[] samples;
    // Construtores
    public WeatherStationMem() {
        count = 0;
        samples = new double[DEFAULT_CAPACITY];
    }
    // Operações/Métodos
    ...
}
```

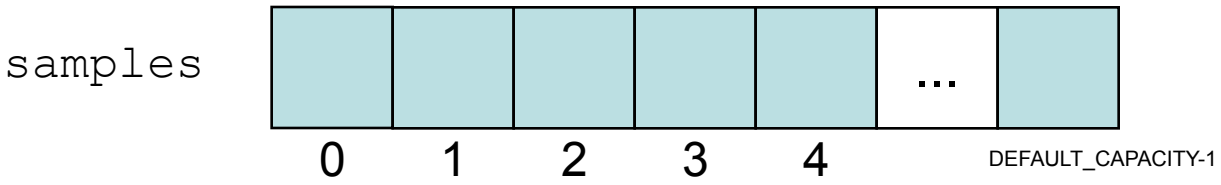
Capacidade, por omissão, do vector de amostras

Criação inicial do vector `samples`, com dimensão ***DEFAULT_CAPACITY***.

Nota: Os índices válidos para um vector de dimensão N são: 0, 1, 2, ..., N-1

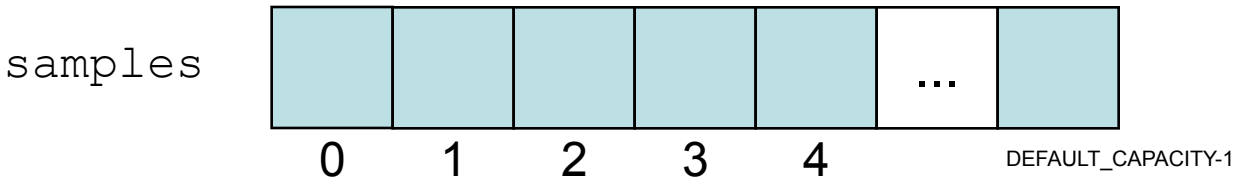
Método `sampleTemperature`

- Como vamos guardar os valores das amostras no vector?



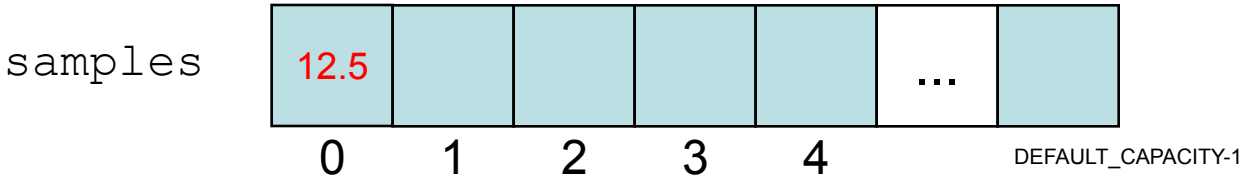
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)



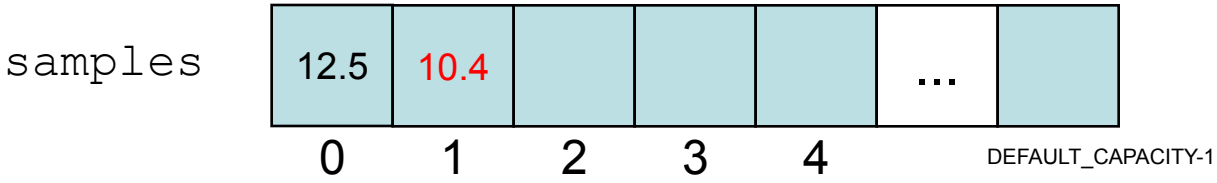
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)



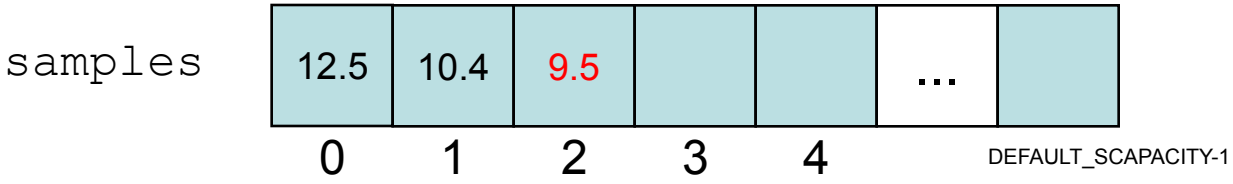
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)



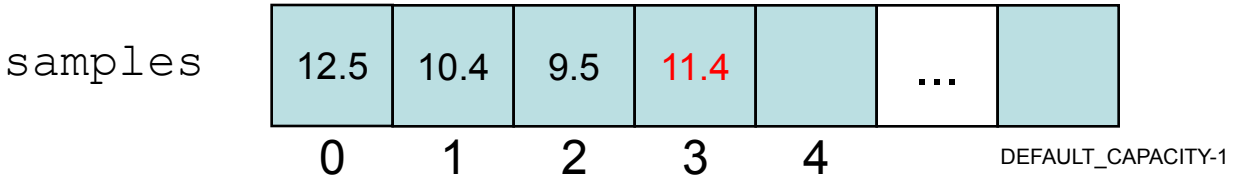
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)



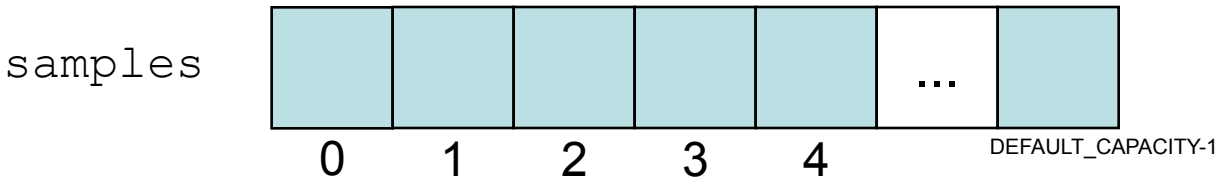
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)



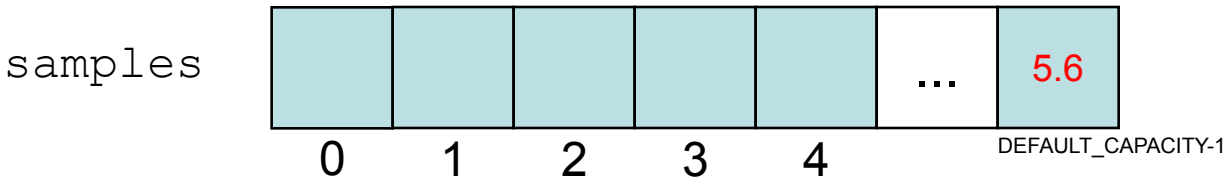
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)



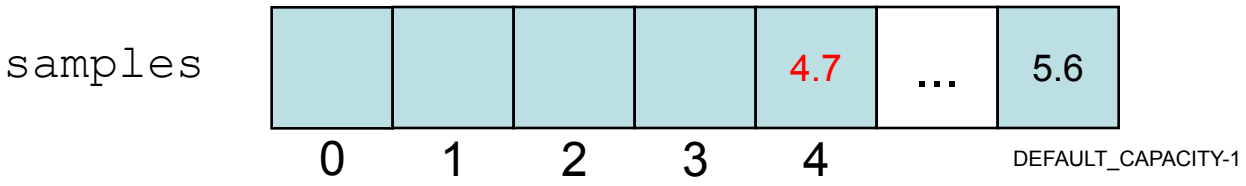
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)



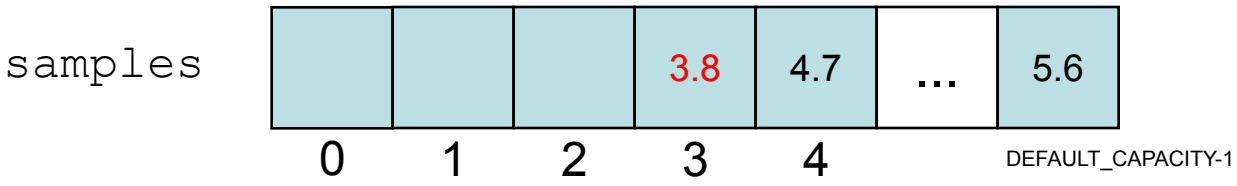
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)



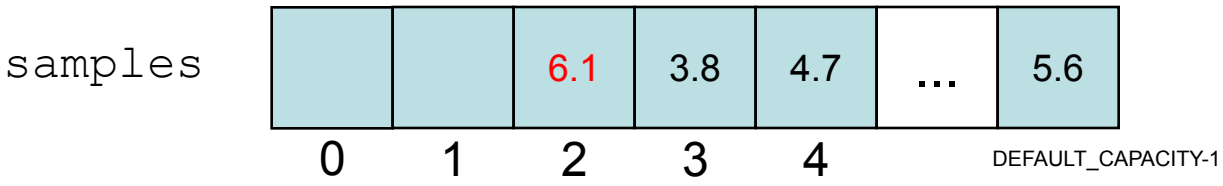
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)



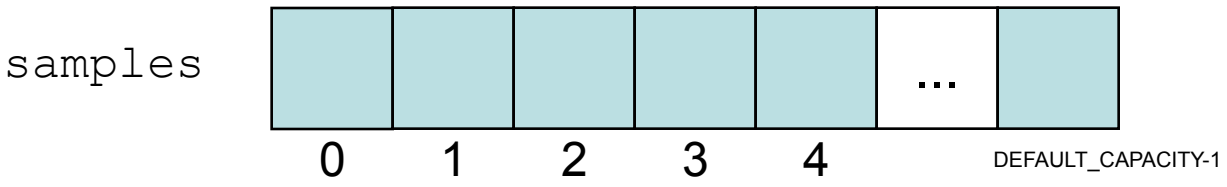
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)



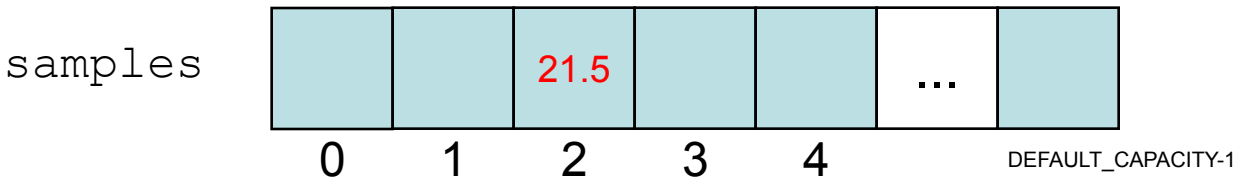
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)
  - de forma aleatória



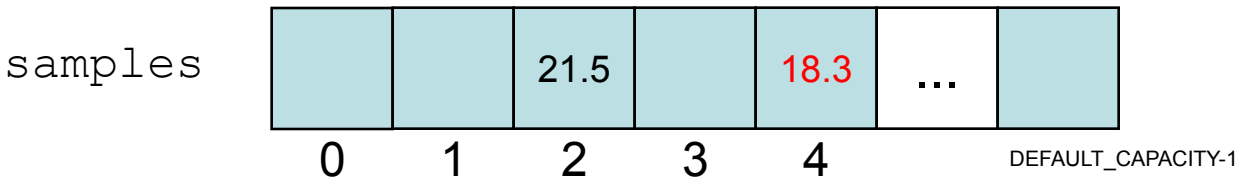
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)
  - de forma aleatória



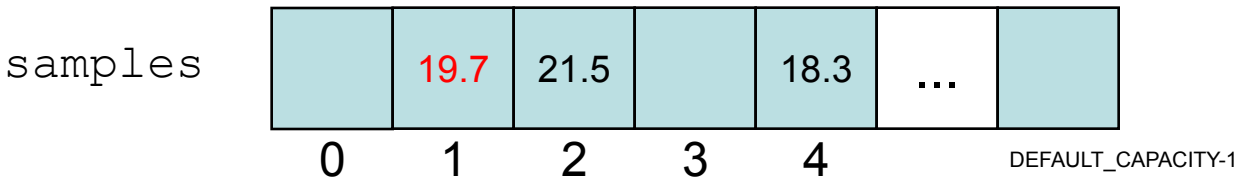
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)
  - de forma aleatória



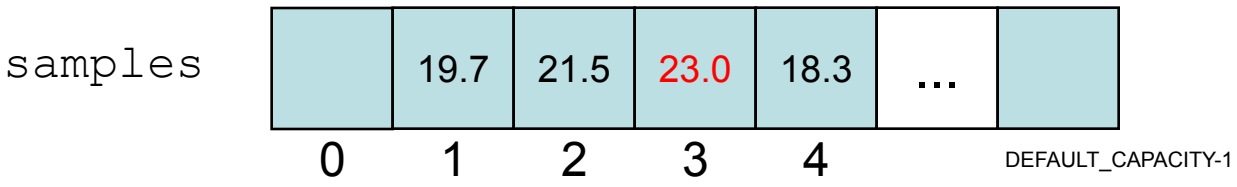
Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)
  - de forma aleatória



Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)
  - de forma aleatória



Vectores

- Um vector pode ser preenchido de diversas formas:
  - por índice crescente (do princípio para o fim)
  - por índice decrescente (do fim para o princípio)
  - de forma aleatória
  - ou de qualquer outra forma que se possa imaginar!

Método `sampleTemperature`

- Como queremos preencher o nosso vector de amostras?

Método `sampleTemperature`

- Como queremos preencher o nosso vector de amostras?
  - No nosso caso queremos preencher **do princípio para o fim**, para **manter a ordem** pela qual as temperaturas vão sendo registadas.
  - Isso é que é coerente com a operação
    `double getSample(int i)`
    que serve - precisamente - para consultar a i-ésima amostra.

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
   samples[count++] = temp;
}
```

Método equivalente a este:

Notas:

Os índices variam entre 0 e ***DEFAULT_CAPACITY***-1

```
public void sampleTemperature(double temp) {
   samples[count] = temp;
   count++;
}
```

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
    samples[count++] = temp;
}
```

Notas:

Os índices variam entre 0 e ***DEFAULT_CAPACITY***-1

Na primeira vez, o valor é guardado na posição 0 e `count` fica com o valor 1

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
    samples[count++] = temp;
}
```

Notas:

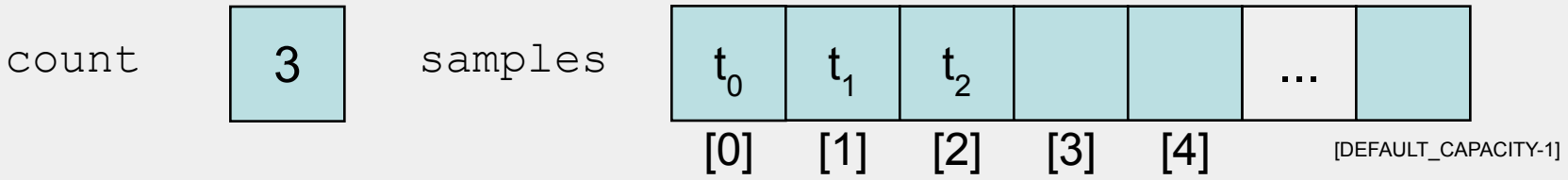
Os índices variam entre 0 e ***DEFAULT_CAPACITY***-1

Na primeira vez, o valor é guardado na posição 0 e `count` fica com o valor 1

Em cada instante, `count` indica o índice da primeira posição livre do vector.

Estação Meteorológica com Memória
(Método `sampleTemperature`)

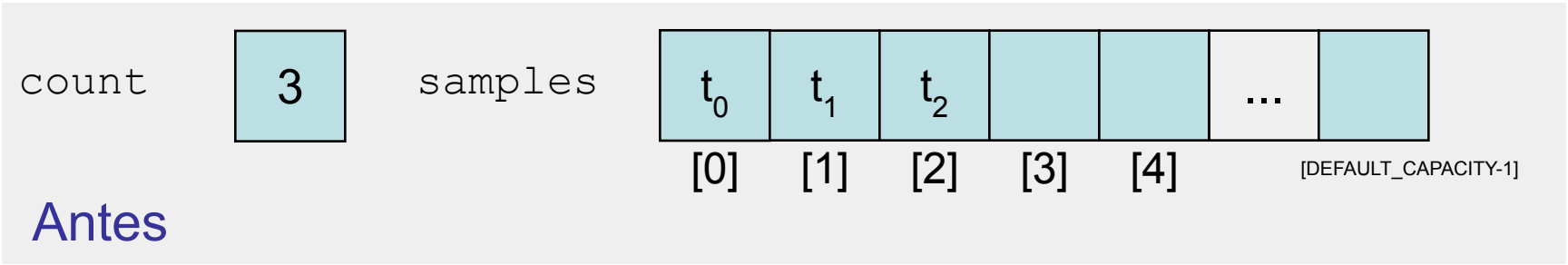
```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
    samples[count++] = temp;
}
```

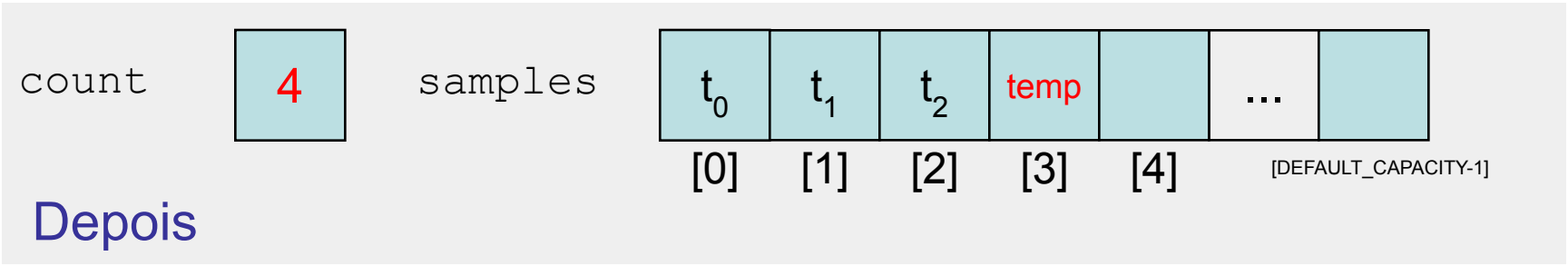


Antes

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
    samples[count++] = temp;
}
```





Estação Meteorológica com Memória (Método `numberOfSamples`)

- E como vamos programar o método `numberTemperatures`?

- É fácil! Até já temos uma variável que serve como contador de amostras na nossa classe!

```
public int numberTemperatures() {
    return count;
}
```

Para além de indicar o índice da primeira posição livre do vector, `count` indica ainda o número de amostras de temperaturas registadas!

Estação Meteorológica com Memória (Método `getSample`)

- E como vamos programar o método `getSample`?
- É fácil! Basta devolver o elemento da posição `i-1` no vector `samples`
  - Recorde que os índices começam em `0`. Deste modo, o primeiro elemento está na posição `0`, o segundo na posição `1`, e assim sucessivamente.

```
//pre: i > 0 && i <= numberTemperatures()
public double getSample(int i){
   return samples[i-1];
}
```

Estação Meteorológica com Memória
(Primeiro resumo)

```
public class WeatherStationMem {
    private static final int DEFAULT_CAPACITY = 100;
    private int count;
    private double[] samples;

    public WeatherStationMem() {
        count = 0;
        samples = new double[ DEFAULT_CAPACITY];
    }

    public void sampleTemperature(double temp) {
        samples[count++] = temp;
    }

    public int numberTemperatures() {
        return count;
    }

    //pre: i>0 && i<=numberTemperatures()
    public double getSample(int i) {
        return samples[i-1];
    }

    ...
}
```

Estação Meteorológica com Memória

- Vamos agora programar as restantes operações da classe `WeatherStationMem`:
  - `public double getAverage()`
  - `public double getMinimum()`
  - `public double getMaximum()`

- Nestas 3 operações:

  Pre: numberTemperatures() > 0

Estação Meteorológica com Memória (Método `getAverage`)

- Como poderemos calcular a média das amostras?

Estação Meteorológica com Memória (Método `getAverage`)

- Como poderemos calcular a média das amostras?
  - Somando todas as amostras registadas no vector e dividir, no final, pelo número de amostras...

Estação Meteorológica com Memória (Método `getAverage`)

- Como poderemos calcular a média das amostras?
  - Somando todas as amostras registadas no vector e dividir, no final, pelo número de amostras...
- Como podemos percorrer o vector de modo a extrair as amostras?

Estação Meteorológica com Memória (Método `getAverage`)

- Como poderemos calcular a média das amostras?
  - Somando todas as amostras registadas no vector e dividir, no final, pelo número de amostras...
- Como podemos percorrer o vector de modo a extrair as amostras?
  - Com um *ciclo*! Começamos no primeiro índice (0) e terminamos na última posição ocupada: (`count-1`)!
- Precisamos de duas variáveis auxiliares (uma para acumular as somas; outra para percorrer o vector)
- A condição do ciclo é não termos ainda chegado à última posição

Estação Meteorológica com Memória
(Método `getAverage`)

```
//pre: numberTemperatures() > 0
public double getAverage() {

   double sum=0;
   for(int i=0;i<count;i++) {
      sum += samples[i];
   }
   return sum/count;
}
```

A variável `i` vai servir para indexar o vector.

somatório das temperaturas

O ciclo é executado para valores de `i` inteiros, iniciando-se em `0` e enquanto forem inferiores a `count`: logo, de `0` até `count-1` !

Estação Meteorológica com Memória (Método `sampleTemperature` melhorado)

- E o que é que acontece quando se registarem mais amostras do que o comprimento do vector?

Estação Meteorológica com Memória (Método `sampleTemperature` melhorado)

- E o que é que acontece quando se registarem mais amostras do que o comprimento do vector?
- O método `sampleTemperature`, tal como se encontra neste momento, tentará o acesso a um índice *inválido* do vector.

```
public void sampleTemperature(double temp){
    samples[count++] = temp;
}
```

Se o vector tiver comprimento igual a `N`, `count` valerá `N` quando o vector encher...

...nesse caso `samples[N]` não existe!!!

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
   if ( isFull() )
      resize();
   samples[count++] = temp;
}

//Verifica se a estacao esta cheia
private boolean isFull() {
   return count == samples.length;
}
```

O método `resize` é um método auxiliar, que vamos criar já a seguir, e que permite "esticar" o tamanho do vector, de modo a conseguirmos armazenar mais elementos.

para um qualquer vector `a`, `a.length` é uma propriedade do mesmo que informa acerca do seu comprimento.

Como fazer sem usar length?

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
// Constante de crescimento do vector
public static final int GROWTH = 2;
...
//Fazer "crescer" o vector; pre: isFull()
private void resize() {
  double[] tmp = new double[GROWTH*samples.length];
```

Como não é possível aumentar o tamanho de um vector começamos por criar um novo, com o dobro do tamanho do antigo!!!

O factor de crescimento tem de ser maior que 1. Escolhemos um factor inteiro para que o tamanho resultante seja um inteiro. Escolhemos o menor inteiro para que o vector vá crescendo apenas o necessário. Finalmente, optamos por um factor multiplicativo, em vez de somar uma constante inteira, porque assim o crescimento de cada vez que o método é chamado é proporcional à dimensão do vector.

```
}
```

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
// Constante de crescimento do vector
public static final int GROWTH = 2;
...
//Fazer "crescer" o vector; pre isFull()
private void resize() {
   double[] tmp = new double[GROWTH*samples.length];
   for(int i=0; i < samples.length; i++)
       tmp[i] = samples[i];
}
```

Este ciclo encarrega-se de copiar para o vector `tmp`, todo o conteúdo do vector `samples`.

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
// Constante de crescimento do vector
public static final int GROWTH = 2;
...
//Fazer "crescer" o vector; pre isFull()
private void resize() {
  double[] tmp = new double[GROWTH*samples.length];
  for(int i=0; i < samples.length; i++)
      tmp[i] = samples[i];
  samples = tmp;
}
```

Ou seja, se `v1` e `v2` forem vectores, fazer `v1=v2` faz com que `v1` passe a referir a sequência de elementos do vector `v2`. Neste caso, apesar de `tmp` ser local e desaparecer no final do método, `samples` fica com o seu conteúdo, para ser usado daqui por diante.

Daqui para a frente, o vector `samples` refere-se ao vector que foi criado com o dobro do tamanho e não ao vector inicial...

Estação Meteorológica com Memória
(Método `sampleTemperature`)

```
...
//Registar a amostra temp na estacao
public void sampleTemperature(double temp) {
   if ( isFull() )
      resize();
   samples[count++] = temp;
}
```

Podemos agora executar a instrução que coloca o valor dentro do vector visto estarmos a usar um vector garantidamente com posições vazias!

Operações comuns sobre vectores

- De modo a praticar algumas destas operações adicione à classe `WeatherStationMem` os seguintes métodos:
  - `public int firstIndexOfSample(double temp)`

    devolve o índice da primeira ocorrência da amostra dada, ou -1 caso não exista;
  - `public int lastIndexOfSample(double temp)`

    devolve o índice da última ocorrência da amostra dada ou -1 caso não exista;
  - `public int countSamplesGreaterThan(double temp)`

    devolve o número de amostras maiores que o valor dado;
  - `public int countSamplesLowerThan(double temp)`

    devolve o número de amostras menores que o valor dado.

Operações comuns sobre vectores

- Como permitem estruturar e armazenar dados, sobre os vectores é comum executar-se uma série de tarefas típicas, tais como:
  - copiar um vector para outro
  - pesquisar a existência de um elemento
  - contar elementos que satisfazem um determinado critério
  - inserir um elemento
  - remover um elemento
  - ordenar um vector

Operações comuns sobre vectores

- Para além de programar os métodos em falta:
  - `public double getMinimum()`
    - **Pre:** `numberTemperatures() > 0`
  - `public double getMaximum()`
    - **Pre:** `numberTemperatures() > 0`
- Para testar estas operações modifique o programa principal de modo a:
  - provocar o esgotamento da capacidade do vector inicialmente reservado para guardar as amostras;
  - indicar nas estatísticas finais o índice da amostra de maior valor assim como o índice da amostra de menor valor;
  - indicar a percentagem de temperaturas negativas registadas pela estação.

Operações comuns sobre vectores

- Suponhamos que por vezes ocorrem erros durante a amostragem cujas consequências são:
  - Algumas amostras surgem repetidas
  - Algumas amostras não são registadas
- O que se pode fazer para corrigir?

Operações comuns sobre vectores

- Suponhamos que por vezes ocorrem erros durante a amostragem cujas consequências são:
  - Algumas amostras surgem repetidas
    - Apagar uma amostra errada do vector
  - Algumas amostras não são registadas
    - Inserir uma amostra no vector, no meio de outras

Operações comuns sobre vectores

Apagar uma amostra do vector

```
//pre: pos > 0 && pos <= numberTemperatures()
public void removeFrom(int pos)
```

- Pretende-se retirar a amostra na posição `pos`.

  Está em `pos-1`

- Para não deixar um espaço "vazio", "puxam-se" os restantes elementos à frente uma posição para trás.

  ```
  samples[i] = samples[i+1];
  ```

- Começa-se pelo elemento a seguir ao apagado, avançando um a um até ao fim, para evitar sobreposições.

  `i` vai até `count-2` para não ultrapassarmos o último elemento

  Quando i for `count-2` acedemos a `i+1=count-2+1=count-1`!

Operações comuns sobre vectores

Apagar uma amostra do vector

```
    //pre: pos > 0 && pos <= numberTemperatures()
    public void removeFrom(int pos) {
        int posV=pos-1;//posicao no vector
        for (int i = posV; i < count-1; i++)
            samples[i] = samples[i+1];//"puxa para tras"
        count--; //decrementa contador de amostras
}
```

Operações comuns sobre vectores

Inserir uma amostra no vector, no meio de outras

```
   //pre: pos > 0 && pos <= numberTemperatures()+1
(pode-se inserir a seguir ao último)
   public void insertAt(double temp, int pos)
```

- Se há espaço no vector, “abre-se” lugar na posição pos para lá colocar temp.
- Para abrir espaço, “empurram-se” os restantes elementos uma posição para a frente

Operações comuns sobre vectores

Inserir uma amostra no vector, no meio de outras

```
   //pre: pos > 0 && pos <= numberTemperatures()+1
(pode-se inserir a seguir ao último)
   public void insertAt(double temp, int pos)
```

- Se há espaço no vector, "abre-se" lugar na posição pos para lá colocar temp.
- Para abrir espaço, "empurram-se" os restantes elementos uma posição para a frente (`samples[i+1] = samples[i];`)

  A partir da esquerda?

  Assim copia-se `samples[pos]` para todas as posições até `count-1`

Operações comuns sobre vectores

Inserir uma amostra no vector, no meio de outras

```
//pre: pos > 0 && pos <= numberTemperatures()+1
(pode-se inserir a seguir ao último)
public void insertAt(double temp, int pos)
```

- Se há espaço no vector, "abre-se" lugar na posição pos para lá colocar temp.
- Para abrir espaço, "empurram-se" os restantes elementos uma posição para a frente (`samples[i+1] = samples[i];`)

  A partir da esquerda?

  Assim copia-se `samples[pos]` para todas as posições até `count-1`
- Na verdade, tem de se começar pelo fim para evitar sobreposições.

Operações comuns sobre vectores

Inserir uma amostra no vector, no meio de outras

```
// pre: pos > 0 && pos <= numberTemperatures()+1
public void insertAt(double temp, int pos) {
    int posV=pos-1; //posicao no vector
    if (isFull()) resize();
    openSpace(posV);
    samples[posV] = temp; //coloca temp em posV
    count++; //incrementa numero de amostras
}
```

O método auxiliar `openSpace` é um ciclo que copia todos os elementos uma posição para a direita (`samples[i+1] = samples[i];`),

a partir do último (`count-1`) até `posV` ("liberta" a posição `posV`)

Operações comuns sobre vectores

```
// pre: pos > 0 && pos <= numberTemperatures()+1
public void insertAt(double temp, int pos) {
    int posV=pos-1; //posicao no vector
    if (isFull()) resize();
    openSpace(posV);
    samples[posV] = temp; //coloca temp em posV
    count++; //incrementa numero de amostras
}
```

O método auxiliar `openSpace` é um ciclo que copia todos os elementos uma posição para a direita (`samples[i+1] = samples[i];`),

a partir do último (`count-1`) até `posV` ("liberta" a posição `posV`)

```
// pre: pos >= 0 && pos < numberTemperatures()
private void openSpace(int posV) {
    for (int i = count-1; i >= posV; i--)
        samples[i+1] = samples[i];
}
```

E se quiser todas as temperaturas acima de um determinado limiar?

```
//@return objecto WeatherStationMem só com as temperaturas
que cumprem o critério
//pre: numberTemperatures() > 0
public WeatherStationMem temperaturesOver(double temp) {
    WeatherStationMem result = new WeatherStationMem();

     /* Aqui, temos de copiar para o vector no objecto
     result as temperaturas acima do limiar temp,
     preservando a ordem das temperaturas no vector
     original. */

    return result;
}
```

E se quiser todas as temperaturas acima de um determinado limiar?

```
//@return objecto WeatherStationMem só com as temperaturas
que cumprem o critério
//pre: numberTemperatures() > 0
public WeatherStationMem temperaturesOver(double temp) {
    WeatherStationMem result = new WeatherStationMem();

     /* Aqui, temos de copiar para o vector no objecto
     result as temperaturas acima do limiar temp,
     preservando a ordem das temperaturas no vector
     original.

     Acção: if (samples[i] > temp) regista samples[i]
     em result */


    return result;
}
```

E se quiser todas as temperaturas acima de um determinado limiar?

```
//@return objecto WeatherStationMem só com as temperaturas
que cumprem o critério
//pre: numberTemperatures() > 0
public WeatherStationMem temperaturesOver(double temp) {
    WeatherStationMem result = new WeatherStationMem();

     /* Aqui, temos de copiar para o vector no objecto
     result as temperaturas acima do limiar temp,
     preservando a ordem das temperaturas no vector
     original.

     Acção: if (samples[i] > temp)
                result.sampleTemperature(samples[i]);
     */

    return result;
}
```

E se quiser todas as temperaturas acima de um determinado limiar?

```
//@return objecto WeatherStationMem só com as temperaturas
que cumprem o critério
//pre: numberTemperatures() > 0
public WeatherStationMem temperaturesOver(double temp) {
    WeatherStationMem result = new WeatherStationMem();
    for (int i = 0; i < this.count; i++)
        if (this.samples[i] > temp)
            result.sampleTemperature(this.samples[i]);
    return result;
}
```

E se quiser ficar só com as temperaturas acima de um determinado limiar?

Cria-se então um vector auxiliar com o tamanho necessário, copiando para lá apenas as temperaturas acima do limiar

Deve-se preencher o novo vector sequencialmente, da esquerda para a direita - usa-se uma variável auxiliar

No fim “coloca-se” o vector auxiliar em `samples`

Deve-se actualizar `count`

E se quiser ficar só com as temperaturas acima de um determinado limiar?

Cria-se então um vector auxiliar com o tamanho necessário, copiando para lá apenas as temperaturas acima do limiar

Preenche-se o novo vector sequencialmente

No fim "coloca-se" o vector auxiliar em `samples` e actualiza-se `count`

```
//pre: numberTemperatures() > 0
public void selectTemperaturesAbove(double temp) {
    //  alocamos espaço suficiente para o "pior" caso
    double[] aux = new double[count];
    int j = 0;
    for (int i = 0; i < count; i++)
        if (samples[i] > temp)
            aux[j++] = samples[i];
    samples = aux;
    count = j;
}
```

E se quiser ficar só com as temperaturas acima de um determinado limiar?

Cria-se então um vector auxiliar com o tamanho necessário, copiando para lá apenas as temperaturas acima do limiar

Preenche-se o novo vector sequencialmente

No fim "coloca-se" o vector auxiliar em `samples` e actualiza-se `count`

```
//pre: numberTemperatures() > 0
public void selectTemperaturesAbove(double temp) {
    double[] aux = new double[countSamplesGreaterThan(temp)];
    int j = 0;
    for (int i = 0; i < count; i++)
        if (samples[i] > temp)
            aux[j++] = samples[i];
    samples = aux;
    count = j;
}
```

Este método percorre o vector duas vezes:

1. Para saber quantas posições deverá associar ao vector aux;
2. Para preencher o vector aux

WeatherStationMem - Programa Principal

Deve terminar quando recebe um valor não numérico - como ?

WeatherStationMem - Programa Principal

Deve terminar quando recebe um valor não numérico - como ?

```
public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        WeatherStationMem ws = new WeatherStationMem();

        // Ler valores
        System.out.print("Proximo Valor: ");
        while (/* É número? */) {
            // Regista valor
        }

        // Listar todos os valores registados
        // ciclo


        }

        // Imprimir estatísticas...

        in.close();
    }

}
```

WeatherStationMem - Programa Principal

Deve terminar quando recebe um valor não numérico - como ?

```
public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        WeatherStationMem ws = new WeatherStationMem();

        // Ler valores
        System.out.print("Proximo Valor: ");
        while (/* É número? */) {
            // Regista valor
        }

        // Listar todos os valores registados
        for(int i=1; i<=ws.numberTemperatures(); i++) {
            System.out.println(i+"a temperatura: "+ws.getSample(i));
        }

        // Imprimir estatísticas...

        in.close();
    }

}
```

WeatherStationMem - Programa Principal

Deve terminar quando recebe um valor não numérico - como ?

```
public class Main {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        WeatherStationMem ws = new WeatherStationMem();

        // Ler valores
        System.out.print("Proximo Valor: ");
        while (in.hasNextDouble()) {
            ws.sampleTemperature(in.nextDouble());
            in.nextLine();
            System.out.print("Proximo Valor: ");
        }

        // Listar Valores
        for(int i=1; i<=ws.numberTemperatures(); i++) {
            System.out.println(i+"a temperatura: "+ws.getSample(i));
        }

        // Imprimir estatísticas...

        in.close();
    }

}
```

Como ficaria com do_while ?