



Orientação a Objetos



Prof. Isaac



Arrays



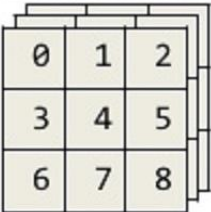
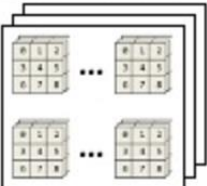


Array

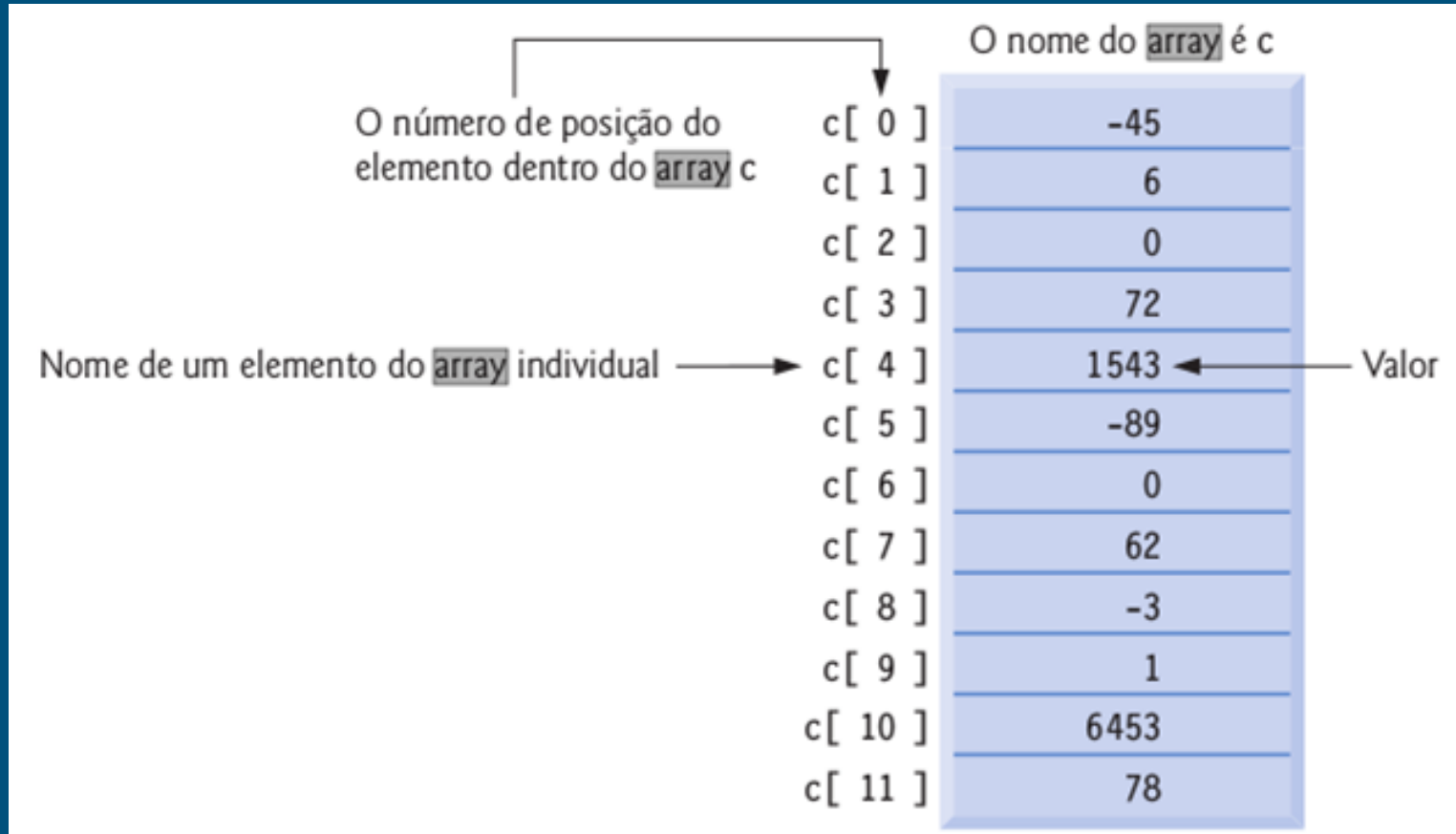
- Coleções de itens de dados relacionados
- Consiste em dados do mesmo tipo
- Arrays são entidades de largura fixa (entidades estáticas)

Array

What is an array?

Dimensions	Example	Terminology
1		Vector
2		Matrix
3		3D Array (3 rd order Tensor)
N		ND Array

Array - Exemplo de 12 elementos



Array - Declaração e criação



```
int n[ 12 ]; // n é um array de 12 inteiros
```

```
int n[ 10 ] = { 0 }; // inicializa elementos do array n como 0
```

```
int n[ ] = { 1, 2, 3, 4, 5 }; // tamanho omitido
```

```
int *n = new int[12]; // dinâmico
```

```
Pessoa n[5]; // array de objetos
```

```
Pessoa n[5] = Pessoa("fulano", "1234"); // todos inicializam  
com o mesmo parâmetro
```

```
int n[ ] = new int[12]; // n é um array de 12 inteiros
```

```
int n[ ] = { 1, 2, 3, 4, 5 }; // tamanho omitido
```

```
Pessoa[ ] n = new Pessoa[5]; // array de objetos  
n[i] = new Pessoa();
```

Array - Declaração e criação

Main.java

```
1  class Main {
2      public static void main(String[] args) {
3
4          int array[]; // declara o array
5
6          array = new int[10]; // cria o array com 10 posições
7
8          System.out.println("Index  Valor");
9
10         for(int count=0; count < array.length; count++)
11             System.out.printf("%3d%5d\n", count, array[count]);
12
13     }
14 }
```

```
javac -classp
ar:target/depe
java -classp
r:target/deper
Index  Valor
0      0
1      0
2      0
3      0
4      0
5      0
6      0
7      0
8      0
9      0
[]
```

Array - Declaração e criação

Main.java

```
1 class Main {
2     public static void main(String[] args) {
3
4         //cria e especifica o valor dos elementos
5         int array[] = {11, 12, 13, 14, 15, 16, 17, 18, 19, 20};
6
7         System.out.println("Index  Valor");
8
9         for(int count=0; count < array.length; count++)
10             System.out.printf("%3d%5d\n", count, array[count]);
11
12     }
13 }
```

```
javac -classpath target/dependencies
java -classpath target/dependencies
Index  Valor
0      11
1      12
2      13
3      14
4      15
5      16
6      17
7      18
8      19
9      20
[]
```


for-each loop

- O Java tem o loop *for-each*, que é usado exclusivamente para percorrer os elementos em um *array*.

- Sintaxe:

```
for (type variable : arrayname) {  
    // code block to be executed  
}
```

- Exemplo:

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};  
for (String i : cars) {  
    System.out.println(i);  
}
```

Arrays Multidimensionais



```
int m[ 2 ][ 2 ] = { { 1, 2 }, { 3, 4 } };
```



```
int m[ ] [ ] = { { 1, 2 }, { 3, 4 } };
```

```
int m[ ] [ ];  
m = new int [ 3 ] [ 4 ];
```

Arrays Multidimensionais



	Coluna 0	Coluna 1	Coluna 2	Coluna 3
Linha 0	<code>a[0][0]</code>	<code>a[0][1]</code>	<code>a[0][2]</code>	<code>a[0][3]</code>
Linha 1	<code>a[1][0]</code>	<code>a[1][1]</code>	<code>a[1][2]</code>	<code>a[1][3]</code>
Linha 2	<code>a[2][0]</code>	<code>a[2][1]</code>	<code>a[2][2]</code>	<code>a[2][3]</code>

Diagram illustrating the structure of a 2D array (matrix) with 3 rows and 4 columns. The array is represented as a grid of cells, each containing a code snippet showing the access syntax: `a[linha][coluna]`.

Labels and arrows indicate the components of the access syntax:

- Subscrito de coluna (Column subscript): Points to the second index (coluna) in the code snippets.
- Subscrito de linha (Line subscript): Points to the first index (linha) in the code snippets.
- Nome do array (Array name): Points to the variable name 'a' in the code snippets.

Exercício

- Faça um algoritmos que armazene 10 números inteiros informados pelo usuário em um array. Exiba como saída o MAIOR número desse array.

ArrayList

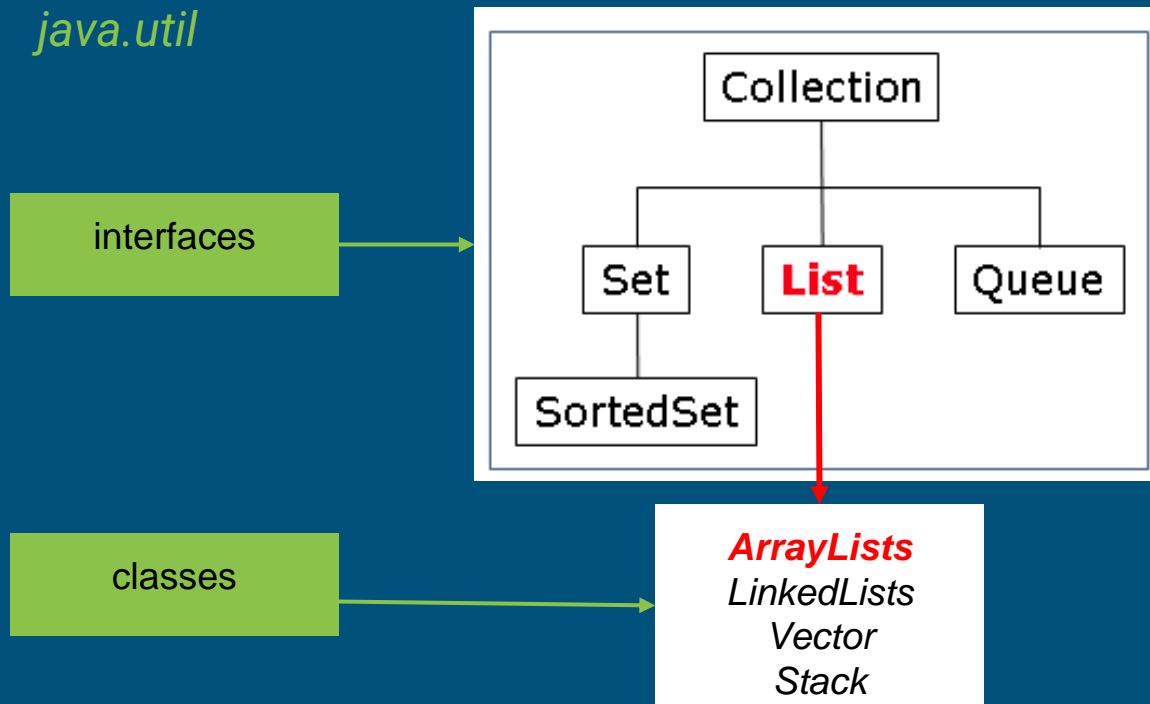


ArrayLists

- *ArrayLists* é parte da estrutura de coleções do Java (*Java Collections*).
- Uma coleção é um objeto que implementa uma estrutura de dados, podendo armazenar referências a outros objetos.
- Não armazena tipos primitivos diretamente, porém podemos utilizar as classes empacotadoras de tipo (Boolean, Integer, Float etc).

Coleções

- As classes e interfaces da estrutura de coleções são membros do pacote *java.util*



ArrayList - Alguns métodos

Método	Método	Descrição
add	add(Object element)	Adiciona um elemento no fim de ArrayList
add	add(int index, Object element)	Adiciona um elemento na posição indicada do ArrayList
clear	clear()	Remove todos os elementos de ArrayList.
contains	contains(Object element)	Retorna true se ArrayList contiver o elemento especificado; senão retorna false
get	get(int index)	retorna o elemento no índice especificado.
indexOf	indexOf(Object element)	Retorna o índice da primeira ocorrência do elemento especificado em ArrayList
remove	remove(Object element)	Remove a primeira ocorrência do valor especificado.
remove	remove(int index)	Remove o elemento no índice especificado.
size	size()	Retorna o número de elementos armazenados no ArrayList.
set	set(int index, Object elemento)	Substitui na posição do index informado pelo elemento especificado..

Fonte: <https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html>

for-each loop

- O Java tem o loop *for-each*, que é usado exclusivamente para percorrer os elementos em um *array*.
- Exemplo:

```
ArrayList<Integer> array = new ArrayList<>();  
  
array.add(10);  
array.add(20);  
array.add(30);  
  
for(int valor: array){  
    System.out.println(valor);  
}
```

Método *forEach* - operação funcional

- O ArrayList tem o método *forEach*, que é usado exclusivamente para percorrer os elementos em um *ArrayList*.
- Exemplo:

```
ArrayList<Integer> array = new ArrayList<>();  
array.add(1);  
array.add(2);  
array.add(3);  
  
array.forEach( elemento -> {  
    System.out.println(elemento);  
});
```

Exercício

- Faça um algoritmos que armazene 10 números inteiros informados pelo usuário em um arrayList. Exiba como saída o MAIOR número desse array.

Exemplo

```
1 import java.util.Scanner;
2 import java.util.ArrayList;
3
4 class Pessoa{
5     public Pessoa(String nome, int idade)
6     {
7         this.nome = nome;
8         this.idade = idade;
9     }
10    public String getNome(){
11        return nome;
12    }
13    public int getIdade(){
14        return idade;
15    }
16
17    private String nome;
18    private int idade;
19};
```

```
21 class TestArrayList
22 {
23     public static void main(String args[])
24     {
25         Scanner entrada = new Scanner(System.in);
26
27         ArrayList <Pessoa> p = new ArrayList<>();
28
29         p.add(new Pessoa("Fulano", 43));
30         p.add(new Pessoa("Sicrano", 28));
31         p.add(new Pessoa("Beltrano", 61));
32
33         for (int i = 0; i < p.size(); i++){
34             System.out.println("Nome: " + p.get(i).getNome());
35             System.out.println("Idade: " + p.get(i).getIdade());
36             System.out.println();
37         }
38     }
39 }
```

```
perico@nuc:~/Dropbox/Development/Java$ java TestArrayList
Nome: Fulano
Idade: 43

Nome: Sicrano
Idade: 28

Nome: Beltrano
Idade: 61

perico@nuc:~/Dropbox/Development/Java$
```

Exemplo *for-each*

```
1 import java.util.Scanner;
2 import java.util.ArrayList;
3
4 class Pessoa{
5     public Pessoa(String nome, int idade)
6     {
7         this.nome = nome;
8         this.idade = idade;
9     }
10    public String getNome(){
11        return nome;
12    }
13    public int getIdade(){
14        return idade;
15    }
16
17    private String nome;
18    private int idade;
19};
```

```
21 class TestArrayList
22 {
23     public static void main(String args[])
24     {
25         Scanner entrada = new Scanner(System.in);
26
27         ArrayList<Pessoa> p = new ArrayList<>();
28
29         p.add(new Pessoa("Fulano", 43));
30         p.add(new Pessoa("Sicrano", 28));
31         p.add(new Pessoa("Beltrano", 61));
32
33         for (Pessoa pessoa : p){
34             System.out.println(pessoa.getNome());
35             System.out.println(pessoa.getIdade());
36             System.out.println();
37         }
38     }
39 }
```

```
perico@nuc:~/Dropbox/Development/Java$ javac TestArrayList.java
perico@nuc:~/Dropbox/Development/Java$ java TestArrayList
Nome: Fulano
Idade: 43

Nome: Sicrano
Idade: 28

Nome: Beltrano
Idade: 61

perico@nuc:~/Dropbox/Development/Java$
```

iteradores (*iterators*)

- As coleções em Java possuem também **iteradores**
- iteradores são **objetos** que permitem ao programador **percorrer uma coleção**
- São usados para apontar para os elementos das coleções

iteradores (*iterators*)

- pacote `java.util.Iterator`

Método	Descrição
<code>hasNext</code>	Returns true if the iteration has more elements.
<code>next</code>	Returns the next element in the iteration.
<code>remove</code>	Removes from the underlying collection the last element returned by this iterator
<code>forEachRemaining</code>	Performs the given action for each remaining element until all elements have been processed or the action throws an exception

Exemplo com *iterator*

```
1 import java.util.Scanner;
2 import java.util.ArrayList;
3 import java.util.Iterator;
4
5 class Pessoa{
6     public Pessoa(String nome, int idade)
7     {
8         this.nome = nome;
9         this.idade = idade;
10    }
11    public String getNome(){
12        return nome;
13    }
14    public int getIdade(){
15        return idade;
16    }
17
18    private String nome;
19    private int idade;
20};
```

```
22 class TestArrayList
23 {
24     public static void main(String args[])
25     {
26         Scanner entrada = new Scanner(System.in);
27
28         ArrayList<Pessoa> p = new ArrayList<>();
29
30         p.add(new Pessoa("Fulano", 43));
31         p.add(new Pessoa("Sicrano", 28));
32         p.add(new Pessoa("Beltrano", 61));
33
34         Iterator<Pessoa> itr = p.iterator();
35
36         while (itr.hasNext()){
37             Pessoa pessoa = itr.next();
38             System.out.println(pessoa.getNome());
39             System.out.println(pessoa.getIdade());
40             System.out.println();
41         }
42     }
43 }
```

perico@nuc:~/Dropbox/Development/Java\$ javac TestArrayList.java

Nome: Fulano
Idade: 43

Nome: Sicrano
Idade: 28

Nome: Beltrano
Idade: 61

perico@nuc:~/Dropbox/Development/Java\$

Método *forEachRemaining*

- Método *forEachRemaining*.

- Exemplo:

```
1  import java.util.ArrayList;
2  import java.util.Iterator;
3
4  class Main {
5      public static void main(String args[]) {
6
7          ArrayList<Integer> array = new ArrayList<>();
8          array.add(1);
9          array.add(2);
10         array.add(3);
11
12         Iterator<Integer> itr = array.iterator();
13         itr.forEachRemaining( elemento -> {
14             System.out.println(elemento);
15         });
16
17     }
18 }
```

```
> javac -classpath . Main.java
> java -classpath . Main
1
2
3
> []
```

Exercício 1

Crie um vetor (com 10 posições) para armazenar sensores a Laser (classe *Laser*):

- Crie a classe *Laser*. Cada Laser deve conter os seguintes atributos: *fabricante*, *alcance*, *precisão* e *medida*
- Depois de criados os 10 objetos Laser, leia as medidas obtidas por cada um deles.

```
//vetor de objetos:  
Laser l[] = new Laser[10];
```

Exercício 2

Escreva um programa em Java que copia um ArrayList para outro ArrayList. Para isso, crie um ArrayList de String e insira nomes. Então copie o ArrayList criado em outro ArrayList.

Exercício 3 - ContaCorrente

- Cria uma classe ContaCorrente com o atributo saldo e os métodos: sacar, depositar e getSaldo.
- Crie um ArrayList com todas as ContasCorrentes e utilize os métodos sacar, depositar e getSaldo de cada uma.

Fim



Material Complementar



vector



vector

- *vector* é parte da *Standard Template Library (STL)* - Biblioteca Padrão de Gabaritos
- STL é uma *biblioteca* padrão do C++
- Descreve containers (sequenciais, adaptadores, associativos) e iteradores, entre outras coisas
- Ela fornece ao desenvolvedor um conjunto de *classes* de uso *genérico*, que podem ser usados com qualquer *tipo de dado* (inclusive próprios)

STL - Containers (alguns exemplos)

- Sequenciais:
 - **vector**: representam arrays que podem mudar de tamanho.
 - **list**: permite inserção em tempo constante e operações de exclusão em qualquer lugar.
 - **deque**: double-ended queue são containers dinâmicos que podem ser expandidos ou encolhidos pelo fim ou pelo começo.

vector

Iterators:

begin	Return iterator to beginning (public member function)
end	Return iterator to end (public member function)
rbegin	Return reverse iterator to reverse beginning (public member function)
rend	Return reverse iterator to reverse end (public member function)
cbegin <small>C++11</small>	Return const_iterator to beginning (public member function)
cend <small>C++11</small>	Return const_iterator to end (public member function)
crbegin <small>C++11</small>	Return const_reverse_iterator to reverse beginning (public member function)
crend <small>C++11</small>	Return const_reverse_iterator to reverse end (public member function)

Capacity:

size	Return size (public member function)
max_size	Return maximum size (public member function)
resize	Change size (public member function)
capacity	Return size of allocated storage capacity (public member function)
empty	Test whether vector is empty (public member function)
reserve	Request a change in capacity (public member function)
shrink_to_fit <small>C++11</small>	Shrink to fit (public member function)

vector

Element access:

operator[]	Access element (public member function)
at	Access element (public member function)
front	Access first element (public member function)
back	Access last element (public member function)
data <small>C++11</small>	Access data (public member function)

Modifiers:

assign	Assign vector content (public member function)
push_back	Add element at the end (public member function)
pop_back	Delete last element (public member function)
insert	Insert elements (public member function)
erase	Erase elements (public member function)
swap	Swap content (public member function)
clear	Clear content (public member function)
emplace <small>C++11</small>	Construct and insert element (public member function)
emplace_back <small>C++11</small>	Construct and insert element at the end (public member function)

Exemplo

```
1#include <iostream>
2#include <algorithm>
3#include <vector>
4
5using namespace std;
6
7class Pessoa {
8public:
9    Pessoa(string nome, int idade)
10    {
11        this->idade = idade;
12        this->nome = nome;
13    }
14    string getNome()
15    {
16        return nome;
17    }
18    int getIdade()
19    {
20        return idade;
21    }
22private:
23    string nome;
24    int idade;
25};
```

```
27int main(){
28
29    vector <Pessoa*> p;
30
31    p.push_back(new Pessoa("Fulano", 62));
32    p.push_back(new Pessoa("Sicrano", 33));
33    p.push_back(new Pessoa("Beltrano", 18));
34
35    for(int i = 0; i < p.size(); i++)
36    {
37        cout << "Nome: " << p.at(i)->getNome() << endl;
38        cout << "Idade: " << p.at(i)->getIdade() << endl;
39        cout << endl;
40    }
41}
```

```
perico@nuc:~/Dropbox/Development/C++$ ./vector
Nome: Fulano
Idade: 62

Nome: Sicrano
Idade: 33

Nome: Beltrano
Idade: 18

perico@nuc:~/Dropbox/Development/C++$
```

Lidando com a memória - objetos estáticos

```
1#include <iostream>
2#include <string>
3#include <vector>
4
5using namespace std;
6
7class Pessoa{
8    public:
9        Pessoa(){};
10        string nome;
11        string cpf;
12        Pessoa(string nome, string cpf) : nome(nome), cpf(cpf) {};
13        ~Pessoa(){}
14};
```

```
16int main()
17{
18    vector<Pessoa> p;
19
20    for (long i=0; i<19999999; i++){
21        Pessoa pessoa("Teste", "123456");
22        p.push_back(pessoa);
23    }
24
25    cout << "retirar do fim do vector: " << endl;
26    cin.get();
27    for (long i=0; i<19999999; i++){
28        p.pop_back();
29    }
30
31    cout << "Tamanho atual de p: " << endl;
32    cout << p.size() <<endl;
33
34    cout << "apagar vetor inteiro com clear" << endl;
35    cin.get();
36    p.clear();
37
38    cin.get();
39
40 }
```

Lidando com a memória - objetos estáticos

The image shows a C++ IDE window titled "C:\Users\tetste\Downloads\problema_mem_est.exe". The console output displays the following text:

```
retirar do fim do vetor:  
Tamanho atual de p:  
0  
apagar vetor inteiro com clear
```

The code editor shows the following C++ code:

```
37  
38     cin.get();  
39  
40 }
```

The compilation results window shows the following output:

```
Compilation results...  
-----  
- Errors: 0  
- Warnings: 0  
- Output Filename: C:\Users\tetste\Downloads\problema_mem_es  
- Output Size: 1,86155319213867 MiB  
- Compilation Time: 0,91s
```

On the right, the Windows Task Manager window is open, showing the "Memória" (Memory) tab. The "Uso da memória" (Memory usage) graph shows a sharp increase in memory usage, which is then followed by a sharp decrease. A green arrow points from the text "instanciando vários objetos do tipo Pessoa" (instantiating several objects of type Pessoa) to the rising part of the graph. Another green arrow points from the text "retirando do vector com pop_back()" (removing from the vector with pop_back()) to the falling part of the graph. The "Composição da memória" (Memory composition) section shows the following values:

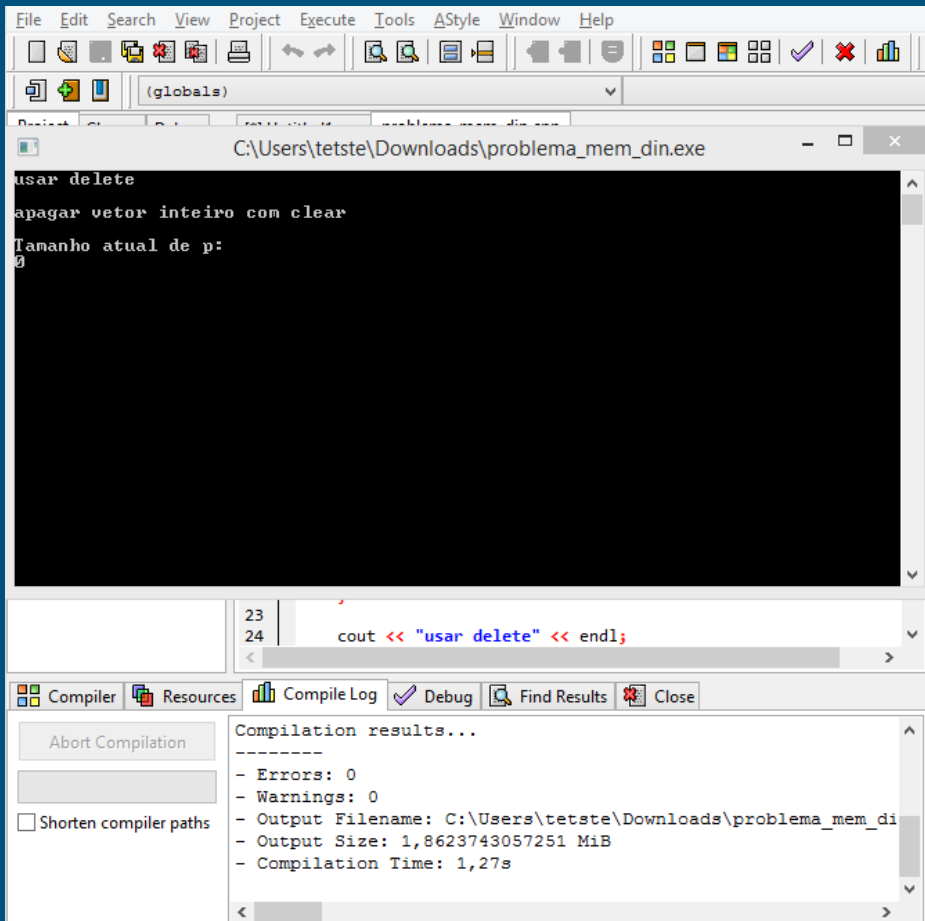
Em uso	Disponível
1,3 GB	2,6 GB
Confirmado	Em cache
1,9/5,8 GB	678 MB
Reserva de memória paginável	Reserva de memória não paginável
132 MB	115 MB

Lidando com a memória - objetos dinâmicos

```
1#include <iostream>
2#include <string>
3#include <vector>
4
5using namespace std;
6
7class Pessoa{
8    public:
9        Pessoa(){};
10        string nome;
11        string cpf;
12        Pessoa(string nome, string cpf) : nome(nome), cpf(cpf) {};
13        ~Pessoa(){}
14};
```

```
16int main()
17{
18    vector<Pessoa*> p;
19
20    for (long i=0; i<19999999; i++){
21        Pessoa *pessoa = new Pessoa("Teste", "123456");
22        p.push_back(pessoa);
23    }
24
25    cout << "usar delete" << endl;
26    cin.get();
27    for (long i=0; i<19999999; i++){
28        delete p.at(i);
29    }
30
31    cout << "apagar vetor inteiro com clear" << endl;
32    cin.get();
33    p.clear();
34
35    cout << "Tamanho atual de p: " << endl;
36    cout << p.size() << endl;
37
38    cin.get();
39}
```

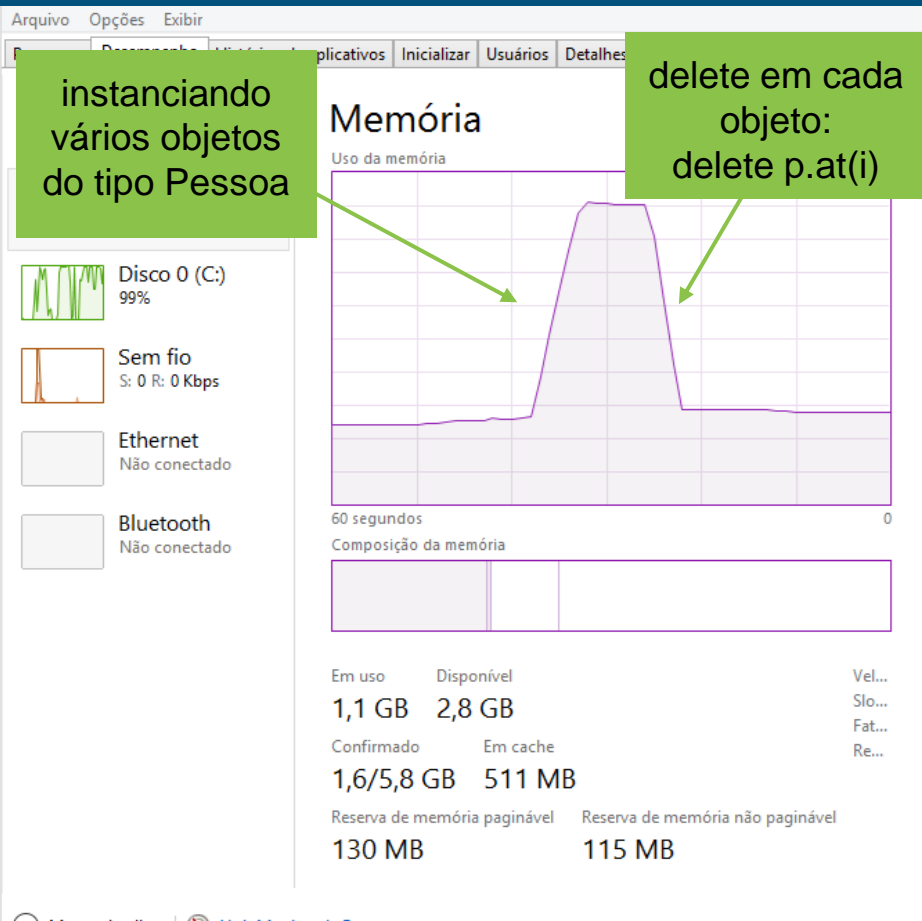
Lidando com a memória - objetos dinâmicos



```
File Edit Search View Project Execute Tools AStyle Window Help
(globalis)
C:\Users\tetste\Downloads\problema_mem_din.exe
usar delete
apagar vetor inteiro com clear
Tamanho atual de p:
0
23
24 cout << "usar delete" << endl;
```

Compilation results...

- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\tetste\Downloads\problema_mem_din.exe
- Output Size: 1,8623743057251 MiB
- Compilation Time: 1,27s



Lidando com a memória - objetos dinâmicos

- O que acontece quando limpamos o *vector* sem deletar os objetos antes?

Lidando com a memória - objetos dinâmicos

- O que acontece se eu limpar o *vector* sem deletar os objetos antes?

