UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS

TECNOLOGIAS DE PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO

Ponteiros ou Apontadores

PROGRAMAÇÃO ORIENTADA A OBJECTOS| Prof. Doutora Frederica Gonçalves

Cofinanciado por:

MADEIRA 14-20

REGIÃO AUTÓNOMA DA MADEIRA

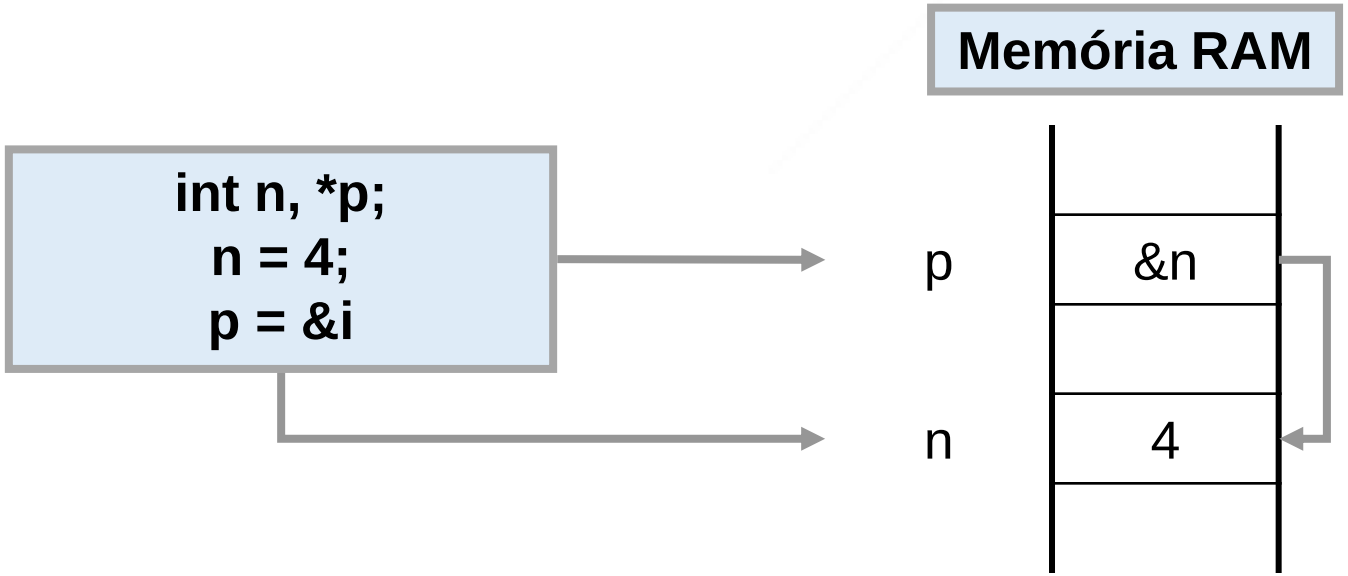
PORTUGAL 2020

UNIÃO EUROPEIA
Fundo Social Europeu

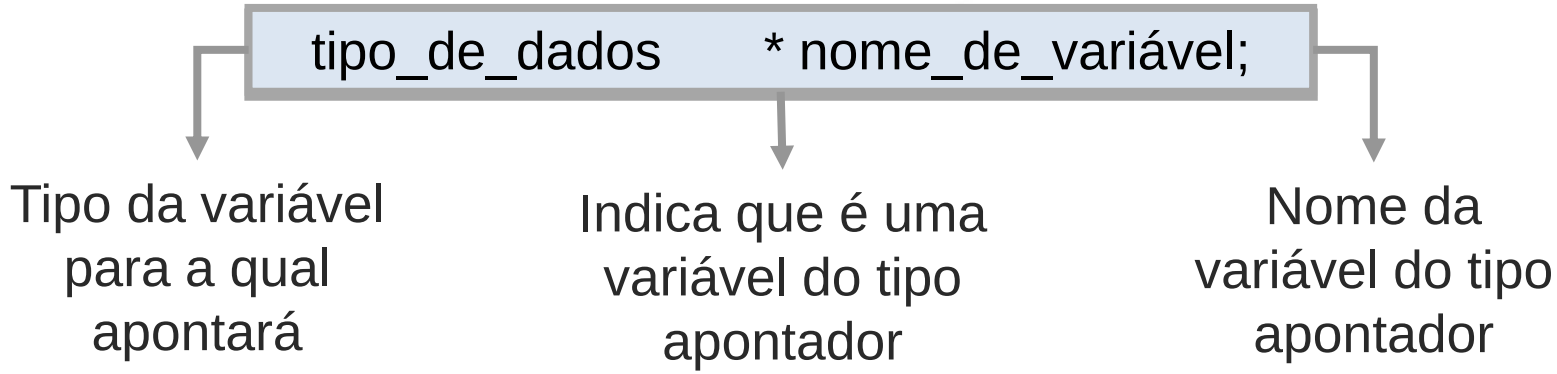
UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

“Todos os dados de um programa residem em algum lugar na memória.”

- **Declaração e utilização de ponteiros:**
  - Um apontador é uma variável como outra qualquer.
  - O seu objectivo é armazenar o endereço de outra variável, o qual é, por sua vez, um número.



- **Declaração e utilização de ponteiros:**
  - Um ponteiro ou apontador (*pointer*) é um tipo especial de variável que é capaz de receber um endereço em memória RAM relativo a uma outra variável (ficando como que a apontar para a própria).
  - Um ponteiro tem de ser declarado antes de poder ser usado.



- **Expressões associadas a um ponteiro declarado (int *p):**

&p	Refere o endereço da própria variável p.
p	Refere o conteúdo de p, que deverá ser o endereço da variável que lhe tiver sido atribuído (uma vez que p é um ponteiro).
*p	Refere o **valor da variável apontada** por **p**.
*	Este sinal assume o papel de **operador de indirecção.**

- **Declaração e utilização de ponteiros:**

Exemplo:

```
int *p;
```

- A variável p está declarada como um ponteiro para variáveis do tipo inteiro.
- Está preparada para receber endereços de variáveis do tipo int.

- **Declaração e utilização de ponteiros:**

  - **& - É o operador ou indicativo de endereço.**
    - Se considerarmos as seguintes declarações:

```
int n, *p;
```

  – Efectuando estas atribuições:

```
n=4, p = &n;
```

Obtemos:

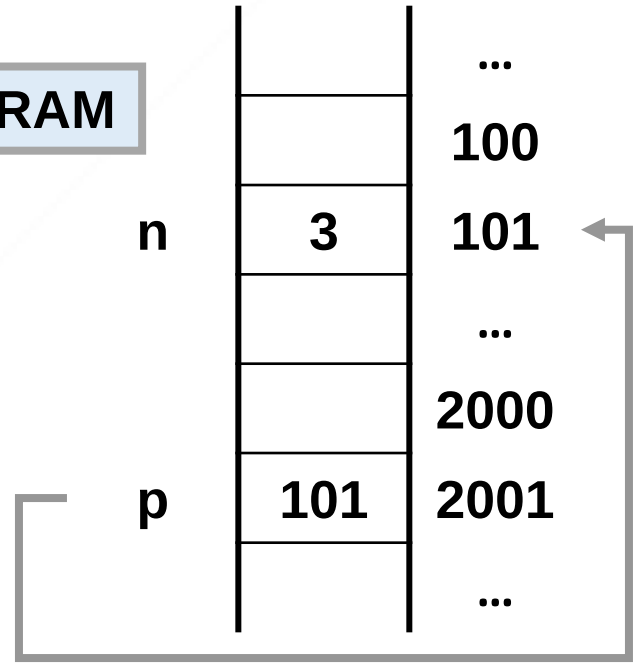
Na variável n o valor 4 e na variável ponteiro recebe o endereço da variável n (p = &n).

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**

Exemplo:

```
int *p; p = &n
```

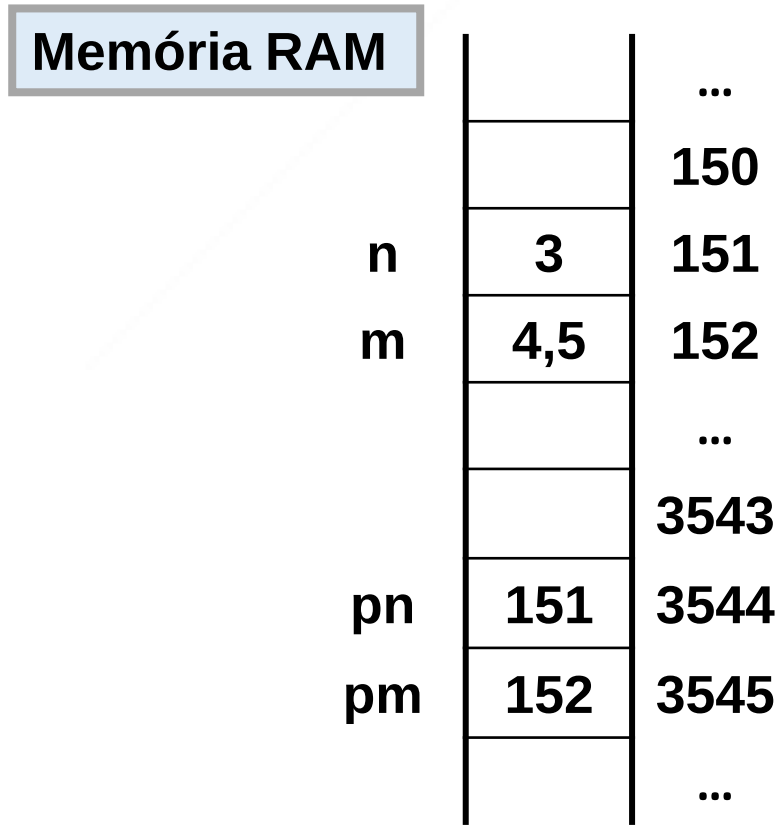


UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

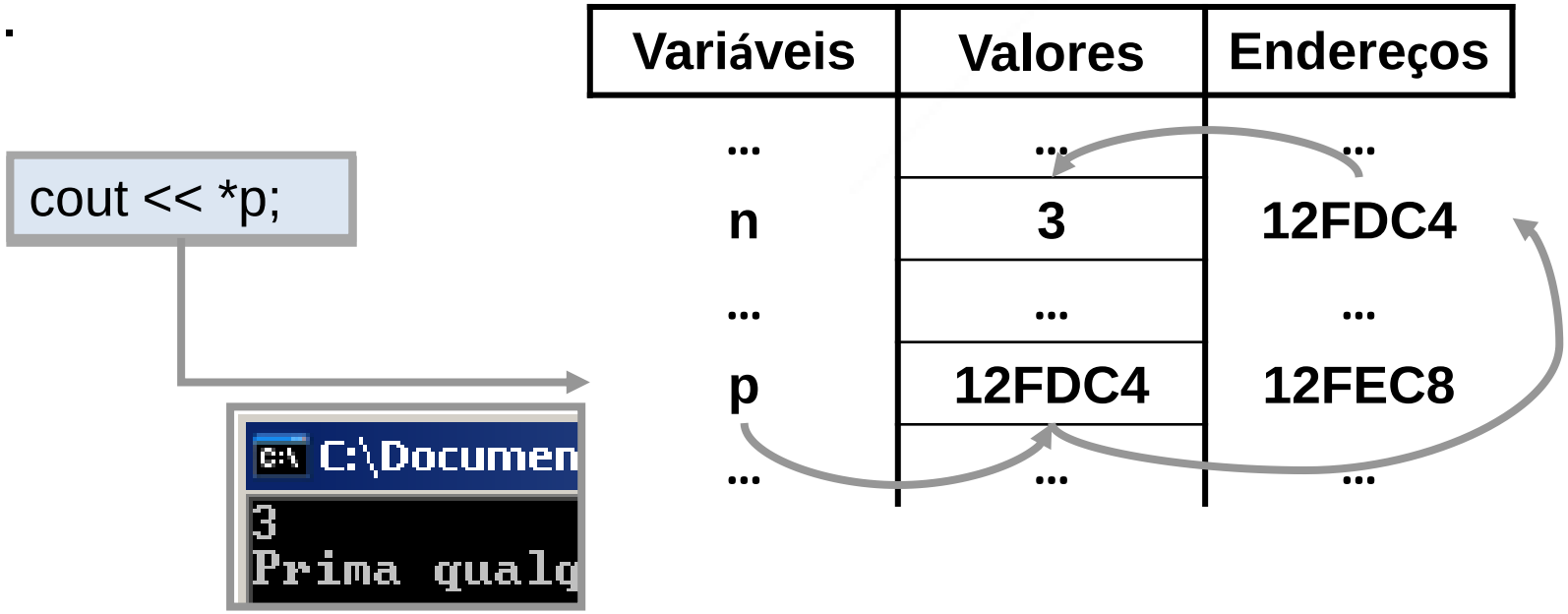
- **Declaração e utilização de ponteiros:**

Exemplo:

```
int n = 3;
float m = 4.5;
int *pn = &n;
float *pm = &m;
```



- **Declaração e utilização de ponteiros:**
  - Representação esquemática de um ponteiro em memória, depois de ser atribuído o endereço da variável n.



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**
  - Atribuição de um valor a uma variável, por apontador:

```
*p = 3;
```

O valor 3, é atribuído, não a p, mas à variável apontada por p.

```
…
 int n, *p;
       n = 4; p = &n;
       cout <<&n <<"\n";
       cout <<n <<"\n";
       cout <<&p <<"\n";
       cout <<p <<"\n";
       cout <<*p <<"\n";
       *p = 3;
       system("pause");
       cout <<p <<"\n" ;
       cout <<n <<"\n";
…
```

- **Declaração e utilização de ponteiros:**
  - Ponteiros e *arrays*:

- Existe uma relação muito estreita entre um ponteiro e um *array*;
- O compilador lida com o nome de um *array* como se fosse um ponteiro.

```
…
        int v [3] = {10, 11, 12};
        cout << &v << "\n" ;
        cout << &v[ 0 ] << "\n";
        cout << &v[ 1 ] << "\n";
        cout << &v[ 2 ] << "\n";
        cout << v[ 0 ] << "\n";
        cout << *v << "\n";
…
```

- **Declaração e utilização de ponteiros:**
  - Ponteiros e *arrays*:
    - O resultado:

```
cout << &v[0];    //escreve o endereço do v[0], que algo do tipo
                  //0x22FF60;
cout << &v        //escreve o mesmo que o de cima, porque esta
                  //a apontar para o inicio do array;
```

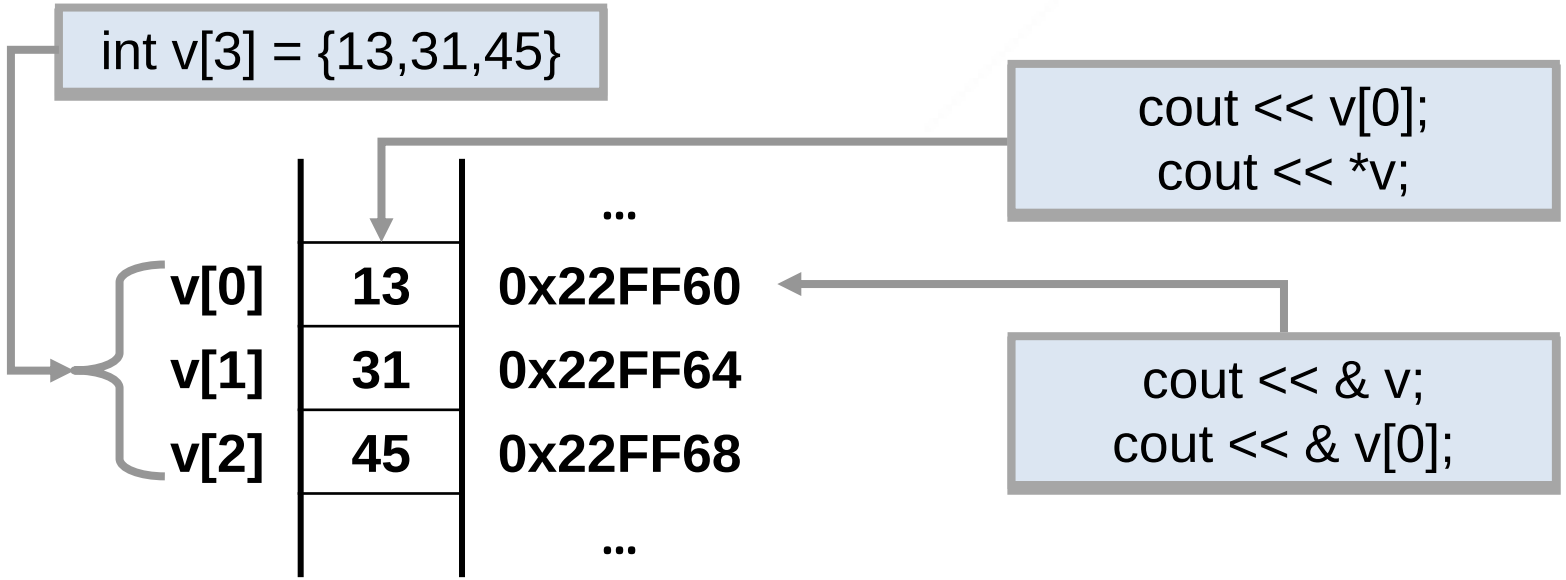
```
cout << &v[1];    //escreve o endereço de v[1], que é 4 bits a cima
                  //do v[0]: 0x22FF64
```

- **Declaração e utilização de ponteiros:**
  - Ponteiros e *arrays*:
    - O resultado:

```
cout << v[0];      //escreve o valor que está no endereço v[0]
```

```
cout << *v[1];     //escreve o valor que está no primeiro endereço
                   //de v
```

- **Declaração e utilização de ponteiros:**
  - Representação esquemática de um *array* e seus endereços de memória RAM:



UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**
  - Apontadores para *arrays*:

```
…
int v [3] = {13, 31, 45};
for (int i=0 ; i<3 ; i++)
cout << v [i] << '\n';
…
```

```
…
int v [3] = {13, 31, 45};
for (int i=0 ; i<3 ; i++)
cout << *(v +i) << '\n';
…
```

"Descubra as diferenças!"

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**
  - Apontadores para *arrays*:

```
…
int v [3] = {13, 31, 45};
for (int i=0 ; i<3 ; i++)
cout << v [i] << '\n';
…
```

```
…
int v [3] = {13, 31, 45};
for (int i=0 ; i<3 ; i++)
cout << *(v +i) << '\n';
…
```

"Descubra as diferenças!"

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**

  - Porque é que dá o mesmo resultado?

    - Se v é um array de i inteiro, então:
      - (v + i) é equivalente ao endereço de v [i];
      - *(v + i) é equivalente ao valor de v [i].

- **Declaração e utilização de ponteiros:**
  - Exemplo:

```
…
int v [3] = {13, 31, 45};
int *p = v;
for (int i=0 ; i<3 ; i++)
cout << *(p + i) << '\n';
…
```

*(p + i) funciona como nos exemplos anteriores, com *(v + i)

- **Declaração e utilização de ponteiros:**
  - Aritmética de ponteiros:
    - Existem operadores aritméticos:
    - Operador de adição: +;
    - Operador de subtracção: -;
    - Operador de incremento: ++;
    - Operador de decremento: --;

*(p + i)?

*(p + 1)?

- **Declaração e utilização de ponteiros:**
  - Exemplo:
    - Cada valor inteiro ocupa na memória 4 *bytes*;
    - Logo, se o endereço de v [0] é, por exemplo: 0x22FF60;
    - O endereço de v [1] será: 0x22FF64;

```
…
    int v [3] = {13, 31, 45};
    cout << &v[0]<<"\n";
    cout << &v[1]<<"\n";
    cout << &v[2]<<"\n";

    int *p = v;
    cout << p << "\n";
    cout << p + 1<< "\n";
    cout << p + 2<< "\n";
…
```

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**
  - Fazendo jogos de incremento e decremento:

```
…
        int v [3] = {13, 31, 45};
        int *p = v;
        cout << p << "\n";
        cout << ++p<< "\n";
        cout << p++<< "\n";
        cout << p << "\n";
…
```

- Se o endereço v [0] for 0x22CD32, qual os resultados das operações de *output*?

- cout << p 0x22CD32;
- cout << ++p 0x22CD36;
- cout <<p++ 0x22CD36;
- cout << p 0x22CD40;

UNIDADE CURRICULAR : PROGRAMAÇÃO ORIENTADA A OBJECTOS

- **Declaração e utilização de ponteiros:**
  - Fazendo jogos de incremento e decremento:

```
…
        int v [3] = {13, 31, 45};
        int *p = v;
        cout << *p << "\n";
        cout << *(p+1)<< "\n";
        cout << *p<< "\n";
        cout << *p++ << "\n";
        cout << *p << "\n";
…
```

- Qual o *output* deste programa?

- cout << *p 10;
- cout << *(p+1) 11;
- cout << *p 10;
- cout << *p++ 10;
- cout << *p 11;

UNIVERSIDADE da MADEIRA

CTeSP CURSOS TÉCNICOS SUPERIORES PROFISSIONAIS