

Algoritmos II



PONTEIROS EM C++

Conceito



- Ponteiro é uma variável que armazena o endereço de memória de um valor, e não o valor.
- Por exemplo: existe uma variável chamada **idade** com o valor **9** e armazenada no endereço **A12A**. Uma variável ponteiro armazenaria o endereço **A12A**, e não o valor 9.
- Um ponteiro pode apontar para os tipos de dados de C++, vetores, funções, strings e estruturas.
- **Declaração (sintaxe):**

```
<tipo do ponteiro> * <nome do ponteiro>;
```

Conceito



- Exemplo:

```
int *ponteiro1, *ponteiro2;  
float *abc;
```

- O tipo do ponteiro deve ser identificado, sendo um dos tipos comuns da linguagem.
- Por exemplo, um ponteiro do tipo double pode guardar endereços de memória de variáveis do tipo double, mas normalmente não pode conter ponteiros para variáveis de outros tipos, como int ou char.

Conceito



- Ao se trabalhar com ponteiros ou variáveis ponteiros se fala em apontar, e não em endereços. Exemplo: se a variável ponteiro denominada `pont1` contém o endereço de memória da variável `var1`, diz-se que `pont1` aponta para `var1`.
- O caracter `&` é um operador de endereço. Diante de uma variável comum produz o endereço dessa variável, ou seja, produz um ponteiro que aponta para a variável.
- O caracter `*`, quando colocado em frente a uma variável ponteiro, referencia o valor contido no endereço de memória armazenado em um ponteiro. É chamado, neste caso, de operador de referência (indireção).

Conceito



Declarada variável
ponteiro do tipo int

Foi atribuído, para a
variável ponteiro
pont1, o endereço
de memória da
variável v

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

int main() {

    int *pont1, v = 5;

    pont1 = &v;
    cout<<endl<<*pont1;
    cout<<endl<<pont1;

    return 0;
}
```

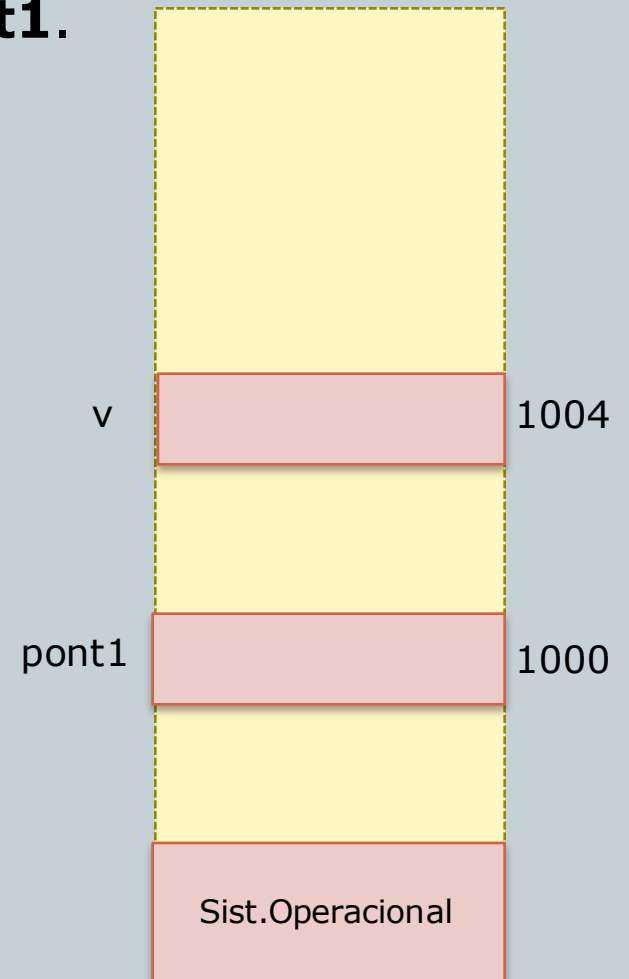
Imprime o valor para
onde a variável ponteiro
pont1 está apontando,
ou seja, irá imprimir 5.

Imprime o endereço de
memória armazenado
na variável pont1, ou
seja, o endereço de
memória da variável v.

Conceito



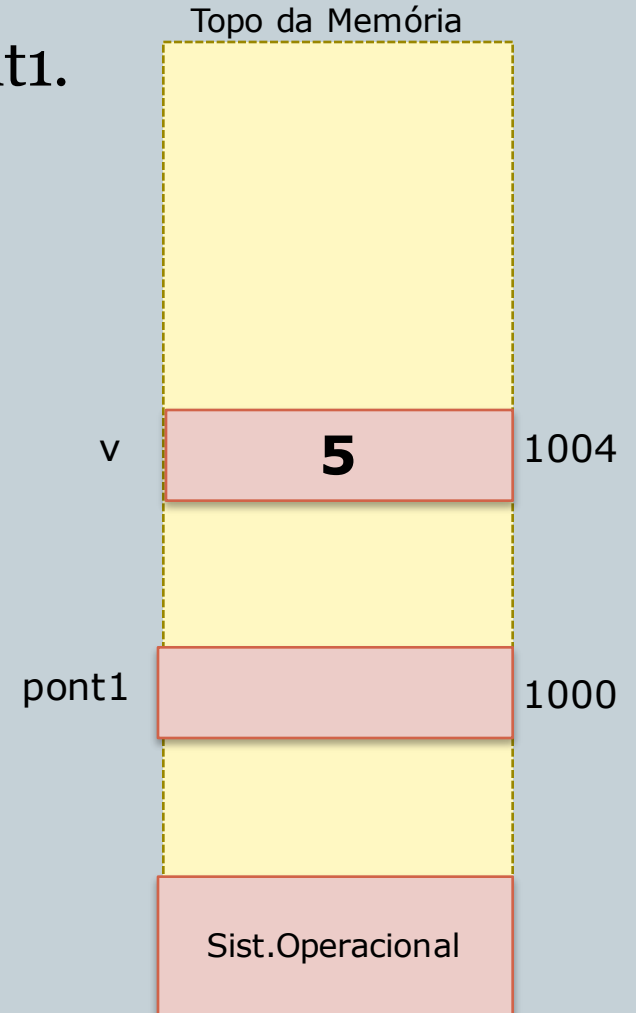
Tem-se a variável **v** e um ponteiro **pont1**.



Conceito



- Tem-se a variável v e um ponteiro pont1 .
- O conteúdo (valor) da variável v é 5.



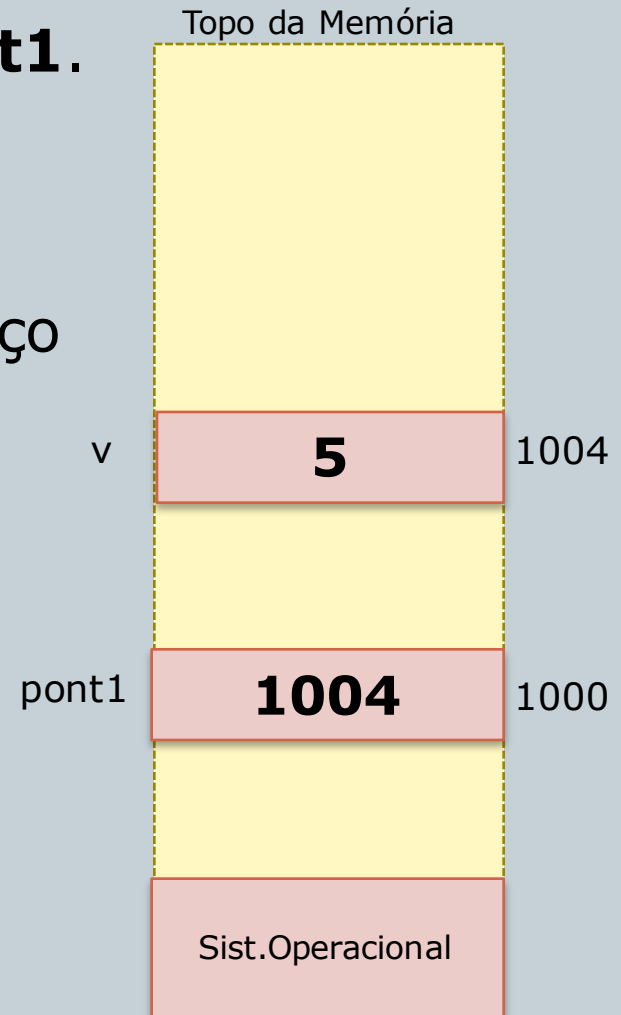
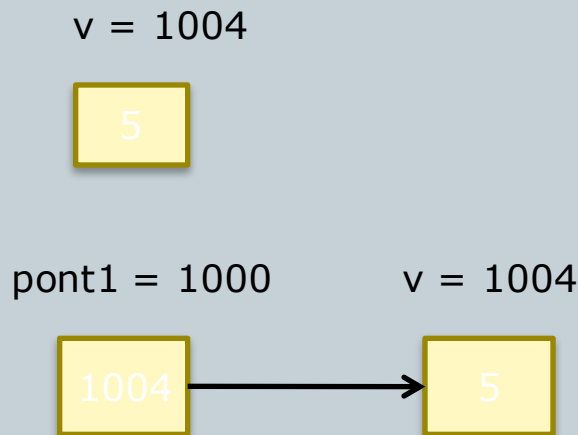
Conceito



Tem-se a variável **v** e um ponteiro **pont1**.

O conteúdo (valor) da variável **v** é **5**.

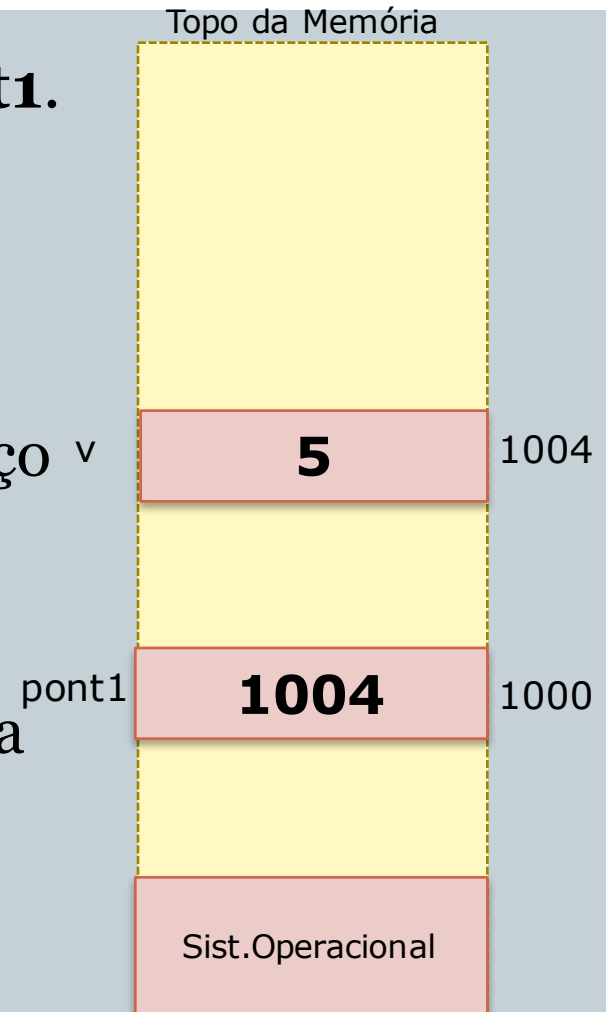
O ponteiro **pont1** aponta para o endereço da variável **v**.



Conceito



- Tem-se a variável **v** e um ponteiro **pont1**.
- O conteúdo (valor) da variável **v** é **5**.
- O ponteiro **pont1** aponta para o endereço **v** da variável **v**.
- Qual é o conteúdo da posição de memória apontada por **pont1**?

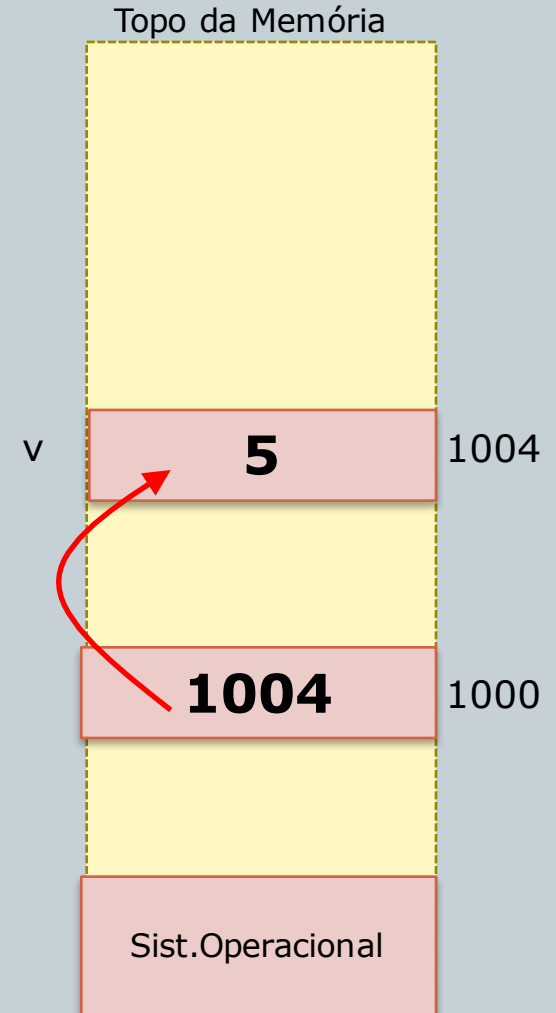


Conceito



- Tem-se a variável **v** e um ponteiro **pont1**.
- O conteúdo (valor) da variável **v** é **5**.
- O ponteiro **pont1** aponta para o endereço da variável **v**.
- Qual é o conteúdo da posição de memória apontada por **pont1**?

[1004] = 5



Conceito



- Pode-se atribuir o valor de uma variável ponteiro para outra variável ponteiro.

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

int main() {

    int *p1, *p2, v = 5, x = 3;

    p1 = &v;
    p2 = &x;

    *p1 = *p2;
    cout<<endl<<*p1;
    cout<<endl<<*p2;
```

```
*p1 = 9;
cout<<endl<<endl<<*p1;
cout<<endl<<*p2;

p2 = p1;
cout<<endl<<endl<<*p1;
cout<<endl<<*p2;

*p2 = 10;
cout<<endl<<endl<<*p1;
cout<<endl<<*p2;

    return 0;
}
```

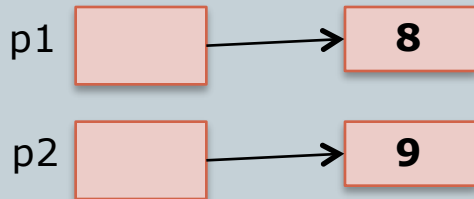
Conceito



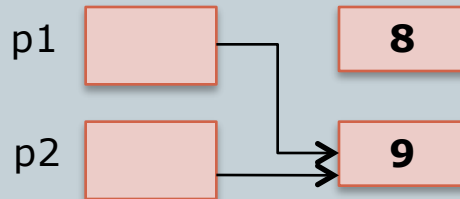
- Mas existem diferenças entre:

p1 = p2

Antes:

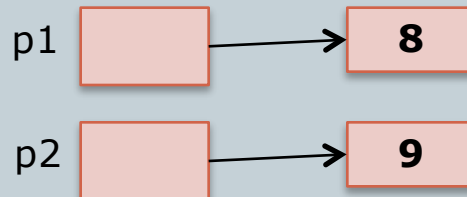


Depois:

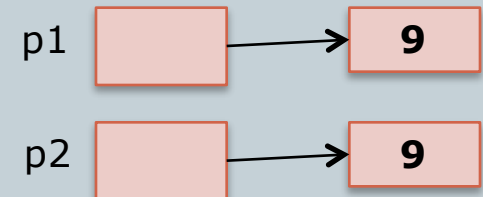


***p1 = *p2**

Antes:



Depois:



Alocação Dinâmica de Memória



- Como um ponteiro pode ser usado para se referir a uma variável, o programa pode manipular variáveis mesmo que essas não tenham identificadores de nomes.
- O operador new pode criar variáveis que não possuem identificadores, mas são referenciadas por meio de ponteiros.
- O operador new realiza o que se chama de alocação dinâmica de memória, e as variáveis criadas por ele são chamadas *variáveis alocadas dinamicamente* ou *variáveis dinâmicas*, por serem criadas e destruídas enquanto o programa é executado.

Exemplo



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

int main () {

    int *p1;

    p1 = new int;

    cin>> *p1;
    *p1 = *p1 + 3;
    cout<<*p1;

    delete p1;
    return 0;
}
```

Exemplo



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

int main () {

    int *p1, *p2;

    p1 = new int;

    *p1 = 42;
    p2 = p1;
    cout<<endl<<endl<<*p1;
    cout<<endl<<*p2;
```

```
*p2 = 53;
cout<<endl<<endl<<*p1;
cout<<endl<<*p2;

p1 = new int;
*p1 = 88;
cout<<endl<<endl<<*p1;
cout<<endl<<*p2;

return 0;
}
```

Teste de Mesa



a) `int *p1, *p2;`

p1 ?

p2 ?

d) `p2 = p1;`

p1 → 42

p2 → 42

g) `*p1 = 88;`

p1 → 88

p2 → 53

b) `p1 = new int;`

p1 → ?

p2 ?

e) `*p2 = 53;`

p1 → 53

p2 → 53

c) `*p1 = 42;`

p1 → 42

p2 ?

f) `p1 = new int;`

p1 → ?

p2 → 53