VI. Vetores de Objetos

Containers simples

Agregação

Vetores: Exercícios

1) Construa um vetor para objetos do tipo Pessoa, no main.
   - como usar o construtor da Pessoa?
   - criar um método preencheDados().
2) Construa um vetor para objetos do tipo Funcionário.
3) Será possível colocar uma Pessoa, um Funcionário e um Professor no mesmo vetor?

Vetores com Alocação Estática

```
(1)
Pessoa(string nome) {
    this->nome = nome;
}
Pessoa p[10] = Pessoa("");

(2)
Pessoa() {}
Pessoa p[10];
```

Vetores com Alocação Estática

```
(3)
Pessoa x[3] = {Pessoa("Ana"), Pessoa("Ciana"), Pessoa("Joana") };

for(Pessoa it: x)
    it.mostra();

(4)
string nomes[] = {"Vinicius", "Igor", "Carlos", "Guilherme",
                        "Filipe"};
Pessoa y[5];

for(int i=0; i<5; i++) {
       y[i] = Pessoa(nomes[i]);
       y[i].mostra();
}
```

Exemplo

```
int main() {
	Pessoa p1("Sara");
	Pessoa p2("Cleberson");
	Pessoa p3("Luis Alberto");

	Pessoa p[10];
	p[0] = p1;
	p[1] = p2;
	p[2] = p3;

	for(Pessoa it: p)
		if(it.getNome()!="")
			it.mostra();
}
```

Vetores

- Declaração

```
Pessoa pessoas[10];
```

- Inicialização padrão

```
Pessoa pessoas[10] = Pessoa(1,“nome”,Data(1,1,1900));
```

- Loops

```
for(int i=0; i<10; i++)
   pessoas[i].mostra();
```

- Foreach loops (C++11)

```
for(Pessoa p: pessoas)
   p.mostra();
```

Habilitando C++11

DevCpp

- Ferramentas
- Opções do Compilador
- Configurações
- Geração de Código
- Padrão da Linguagem (-std)
- Marque a opção “Iso C++ 11”

Code::Blocks

- Settings
- Compiler
- Compiler settings
- Compiler flags
- Marque a opção “Have g++ follow the C++11 ISO C++ language standard [-std=c++11]”

Polimorfismo em Herança

- Posso atribuir um Funcionário numa Pessoa, porque ele também é uma Pessoa (herança).

- Mas não posso atribuir uma Pessoa num Funcionário, apesar do funcionário ter uma parte pessoa.

POLIMORFISMO

Polimorfismo

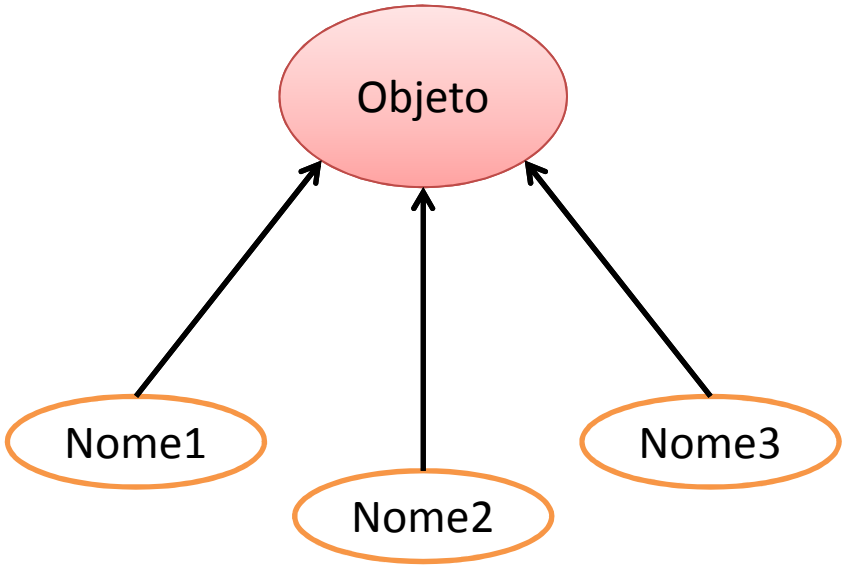
- Em C++ indica a habilidade de uma única instrução chamar diferentes funções e portanto assumir formas diferentes.
- Nos nossos exemplos:
  - Um Funcionário também é uma Pessoa
  - Um Gerente também é um Funcionário
- Isto permite que eu possa acessar o objeto por uma "interface" diferente.

Tipos de Polimorfismo

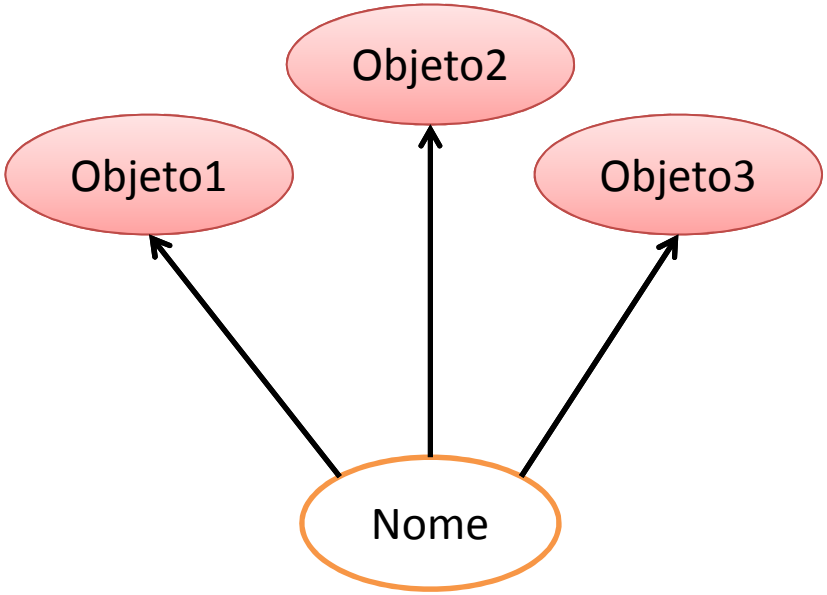
- Existem quatro tipos de polimorfismo que a linguagem pode ter
- Universal
  - Inclusão: um ponteiro da classe mãe pode apontar para uma instância de uma classe filha
  - Paramétrico: se restringe ao uso de templates (C++, por exemplo) e generics (C#/Java)
- Ad-Hoc
  - Sobrecarga: duas funções/métodos com o mesmo nome mas assinaturas diferentes
  - Coerção: conversão implícita de tipos sobre os parâmetros de uma função

Diferenças!!!

Inclusão



Sobrecarga



Funções virtuais

- O compilador ignora o conteúdo do ponteiro p1 e usa o seu tipo para identificar a função-membro a ser chamada.
- Para acessar objetos de diferentes classes usando a mesma instrução, devemos declarar as funções da classe-base que serão reescritas em classes derivadas usando a palavra **virtual**.

- **Resolução dinâmica** : permite que uma instrução seja associada a uma função no momento de sua execução.

Exemplo

```
class Pessoa {
    virtual void mostra();
};

int main() {
    Pessoa p1("Sara");
    Funcionario p2("Cleberson");
    Professor p3("Luis Alberto");

    Pessoa *p[10] = {NULL};
    p[0] = &p1;
    p[1] = &p2;
    p[2] = &p3;

    for(Pessoa *it: p)
        if(it != NULL)
            it->mostra();
}
```

EXERCÍCIOS

Exercícios

- Faça um vetor para a classe Conta e insira e manipule algumas:
  - contas correntes
  - contas poupança
  - contas especiais