



Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: BCC 221 Programação Orientada a Objetos

Trabalho Prático 1

1 Instruções

- O problema deve ser resolvido por meio de um programa em C++ e a STL;
- Utilize o site <http://www.cplusplus.com/> como referência;
- O código-fonte deve estar devidamente comentado;
- Não serão aceitos trabalhos que caracterizem cópia (mesma estrutura e algumas pequenas modificações, por exemplo) de outro ou de códigos da internet;
- Eventualmente, após a entrega dos trabalhos serão marcadas entrevistas com cada um dos alunos para apresentação dos mesmos para o professor.

2 Entrega

- A entrega do código-fonte será feita pelo Moodle.
- Deve ser entregue um zip com:
 - Relatório (descrito ao fim deste texto) conforme o modelo “Prática Modelo de Entrega.docx”;
 - Diagrama UML;
 - Código fonte;
 - Descrição de como compilar e executar o projeto pela linha de comando.

3 Avaliação

- Funcionamento adequado do programa:
 - Códigos que não compilarem e tiverem *warnings* diminuirão a nota;
 - Corretude (independente se gerado por IDE ou manualmente).
- Atendimento ao enunciado do trabalho;
- Comentários;
- Identação do código e boas práticas de programação;
- Boa organização do código fonte em geral;
- Bom uso da STL.

4 Enunciado

O objetivo deste trabalho é a implementação de uma corretora de imóveis. A corretora terá um sistema que gerencia imóveis, o qual deverá ser implementado baseado na *Standard Templates Library* (STL).

A escolha dos contêineres da STL deve ser adequado ao contexto da sua utilização, isto é, a sua escolha deve ser pensada de acordo com suas funcionalidade e os algoritmos de manipulação. O objetivo não é usar o contêiner somente como um armazenador de dados, mas sim utilizar suas funções. Além disso, deve-se priorizar sempre os algoritmos que já estão implementados na STL ao invés de reimplementar alguma função já existente.

O arquivo com os imóveis disponíveis na corretora pode ser encontrado pelo link: <https://drive.google.com/file/d/1VZ5bLm7VgzVXPaxJSU5lBvCmfRODNEdJ/view?usp=sharing>. O arquivo possui 16 linhas, em que cada uma delas indica as informações de um imóvel. Assim que o sistema for iniciado, o arquivo deve ser lido para preencher a coleção. O padrão do arquivo é:

- Linha 1: Tipo do imóvel (apartamento; casa e chácara);
- Linha 2: Valor do imóvel;
- Linha 3: Nome do proprietário do imóvel;
- Linha 4: Rua em que o imóvel está localizado;
- Linha 5: Bairro em que o imóvel está localizado;
- Linha 6: Cidade em que o imóvel está localizado;
- Linha 7: Número do imóvel;
- Linha 8: Quantidade de quartos do imóvel;
- Linha 9: Quantidade de banheiros do imóvel;

À partir da Linha 10, as informações dependerão do tipo de imóvel:

- Casa:
 - Linha 10: Quantidade de andares da casa;
 - Linha 11: A existência ou não de sala de jantar;
- Apartamento:
 - Linha 10: O Andar em que ele está localizado;
 - Linha 11: A taxa de condomínio;
 - Linha 12: A existência ou não de elevador;
 - Linha 13: A existência ou não de sacada;
- Chácara:
 - Linha 10: A existência ou não de salão de festas;
 - Linha 11: A existência ou não de salão de jogos;
 - Linha 12: A existência ou não de campo de futebol;
 - Linha 13: A existência ou não de churrasqueira;
 - Linha 14: A existência ou não de piscina;

4.1 Detalhes do Sistema

O sistema conta com três tipos de imóveis cadastrados: casa, apartamento e chácara. O diagrama UML da Figura 1 indica quais os atributos e métodos dos tipos de imóveis cadastrados. Vale ressaltar que, na base de dados, os atributos booleanos estão representados como 1 (true) ou 0 (false).

O sistema em si será codificado no *main* do programa. Todas as operações descritas devem ser implementadas em forma de função e chamadas no *main*. O *main* deverá conter uma única coleção polimórfica de objetos de casa, apartamento e chácara. A coleção criada deve possibilitar a manipulação e, posteriormente, transformações para as seguintes operações:

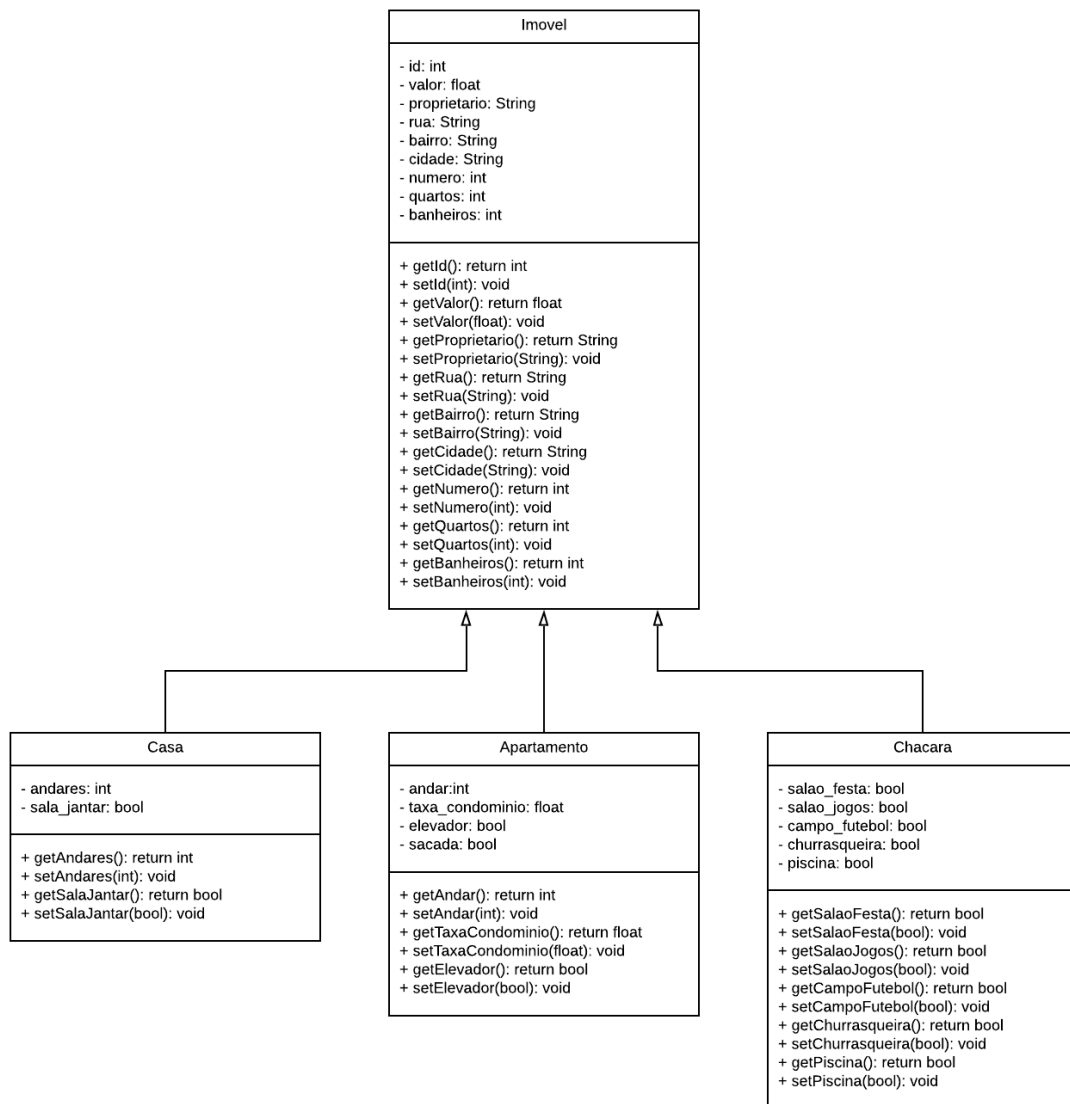


Figure 1: UML do sistema.

1. Sobrecarregar o operador « (saída) para todas as classes. Todas as operações que pedirem somente a exibição do imóvel devem utilizar esse operador. Cada parâmetro especificado deve ser impresso em uma linha. Para separar as informações de dois imóveis diferentes deve-se utilizar uma linha tracejada com o caractere “-”, como

na Figura 2. A impressão deve ser feita de forma estruturada seguindo o seguinte padrão:

- Mostrar o proprietário;
- Mostrar o preço;
- Mostrar o número de quartos;
- Mostrar a rua;
- Mostrar o bairro;
- Mostrar a cidade;
- Mostrar uma característica específica do tipo de imóvel:
 - Apartamento: Indicar a existência de elevador;
 - Casa: Indicar a quantidade de andares;
 - Chácara: Indicar a existência de piscina;

```
-----
Proprietário
  Preço
  Número de Quartos
  Rua
  bairro
  Cidade
  Característica Específica
-----
```

Figure 2: Saída desejada.

2. Dado um proprietário, criar uma função que retorne *true* se houver algum imóvel desse proprietário, ou *false* se não houver. No *main* deve ser exibido se há ou não.
3. Dado um valor, criar uma função que retorne uma única coleção com todos os imóveis (casa, apartamento e chácara) que possuam o valor igual ou abaixo do especificado e exibir o imóvel no *main*, utilizando o operador « sobrecarregado;
4. Dado o número de quartos do imóvel, criar uma função que retorne uma única coleção com todos os imóveis (casa, apartamento e chácara) com o número de quartos igual ou acima do especificado e exibir o imóvel no *main*, utilizando o operador « sobrecarregado;
5. Dado um tipo de imóvel, criar uma função que retorne uma coleção de todos os imóveis deste tipo ordenados pelo seu respectivo valor e exibir os imóveis no *main*, imprimindo todas as suas características específicas;
6. Dada uma cidade, criar uma função que retorne uma única coleção com todos os imóveis (casa, apartamento e chácara) que estejam localizados na cidade especificada. Os imóveis deverão ser exibidos no *main* em ordem decrescente em relação ao seu valor.
7. Dado o nome de um proprietário, criar uma função que retorna um iterador para cada tipo de imóvel apontando para o elemento encontrado. No *main* deve ser mostrado se o imóvel foi encontrado e os dados do mesmo para cada um dos tipos de imóvel (mostrar os dados comuns e específicos do imóvel conforme o item 1).

8. Criar uma função que recebe uma única coleção de imóveis de todos os tipos e que mostre no terminal ou salve em um arquivo (saida.txt) todos os tipos de imóveis. Mostrar os dados comuns e específicos do imóvel conforme o item a). Um argumento passado para a função define qual será a saída. Nesse caso é necessário *downcasting*.

5 Relatório

Deverá ser entregue um relatório descrevendo o porquê da escolha de cada contêiner e algoritmo em cada parte da implementação. Além disso, dado os contêineres presentes na STL, deve ser descrito sucintamente qual o melhor contêiner para cada uma das seguintes situações:

- Acessar uma posição específica de um contêiner;
- Adicionar um elemento e manter somente elementos únicos no contêiner;
- Inserção no final;
- Remoção no final;
- Retornar um valor baseado em uma chave específica (não necessariamente inteiros);
- Inserção no início;
- Remoção no início;
- Busca por um elemento;
- Contêiner com o comportamento de primeiro a entrar e o último a sair;
- Contêiner com o comportamento de primeiro a entrar e o primeiro a sair.