

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO E ENGENHARIA DE  
COMPUTAÇÃO

LEONARDO DALCIN  
PAULO RICARDO RAMOS DA ROSA  
PIETRO DEGRAZIA

## **Jogo Pedagógico em C++**

Relatório apresentado como requisito parcial para  
a obtenção de conceito na Disciplina de Modelos  
de Linguagens de Programação

Prof. Dr. Lucas Mello Schnorr  
Orientador

Porto Alegre  
2018

## SUMÁRIO

<b>1 INTRODUÇÃO .....</b>	<b>3</b>
<b>1.1 Projeto .....</b>	<b>3</b>
<b>1.2 Sobre o jogo .....</b>	<b>3</b>
<b>2 C++17 .....</b>	<b>4</b>
<b>2.1 Origem .....</b>	<b>4</b>
<b>2.2 Programa .....</b>	<b>4</b>
<b>2.3 Objetos .....</b>	<b>4</b>
<b>2.4 Classes .....</b>	<b>5</b>
<b>2.5 Classes Abstratas .....</b>	<b>5</b>
<b>2.6 Polimorfismo .....</b>	<b>6</b>
<b>2.7 Memória .....</b>	<b>6</b>
<b>3 IMPLEMENTAÇÃO .....</b>	<b>7</b>
<b>3.1 Orientação a Objetos .....</b>	<b>7</b>

## **1 INTRODUÇÃO**

Este trabalho tem como objetivo a implementação em C++ de um jogo pedagógico que visa ensinar diversos conceitos da cadeira de Modelos de Linguagens de Programação em diferentes níveis de profundidade. O programa permitirá que o usuário compartilhe seus resultados e confira respostas de outros colegas, assim como explicações e links para referências sobre o assunto.

### **1.1 Projeto**

A proposta do projeto consta em, dada uma linguagem de programação escolhida pelo grupo dentre as pré-selecionadas pelo professor, desenvolver uma aplicação com duas implementações: uma puramente orientada a objetos e outra puramente funcional, contendo alguma forma de paralelismo. C++17 será a linguagem utilizada para ambas as versões.

### **1.2 Sobre o jogo**

O projeto será um jogo de perguntas e respostas sobre a disciplina de Modelos de Linguagem de Programação, aos moldes dos programas "Show do Milhão" e "Quem quer ser um milionário?", com a objetivo de ajudar o estudante na disciplina. O jogo contará com explicações sobre o conteúdo das perguntas e poderá ser feito a consulta das respostas dos outros jogadores.

## 2 C++17

### 2.1 Origem

A linguagem C++ foi introduzida inicialmente em 1985 por Bjarne Stroustrup, herdando as características da linguagem C mas adicionando o conceito de Orientação a Objetos. Desde então, passou por diversas atualizações, estando hoje padronizada na versão ISO/IEC 14882:2017, informalmente conhecida como C++17.

### 2.2 Programa

Programas em C++ podem conter valores, objetos, referências, funções, enumeradores, tipos, membros de classes, templates, especializações de template e namespaces. Essas entidades são criadas através de declarações, que associam a entidade com um nome e define suas propriedades. A declaração que define todas as propriedades para usar uma entidade é chamada de definição.

Nomes encontrados em um programa são associados com a declaração em que aparecem. Eles somente são válidos no escopo em que foram declarados. Alguns nomes, dependendo da linkagem, podem referenciar entidades que aparecem em um escopo diferente.

### 2.3 Objetos

Programas em C++ criam, destroem, referenciam, acessam e manipulam objetos. Um objeto em C++ é uma região de armazenamento que possui:

- tamanho(sizeof);
- requisito de alinhamento(aligned);
- duração de alocação(automática, estática, dinâmica, local);
- duração de vida(baseada na alocação ou temporária);
- tipo;
- valor(pode ser indeterminado);

- nome(opcional).

Objetos são criados com definições, expressões new, expressões throw, por manipulação de membros de uma union e onde objetos temporários são necessários.

## 2.4 Classes

Classes e estruturas são tipos definidos pelo usuário, usando especificadores de classe, que devem aparecer na sequência a seguir.

- Chave de classe - pode ser class ou struct, a única diferença entre elas é o acesso padrão, para estruturas é acesso a membro, e para classes é a membro da classe base.
- Atributos - sequência opcional de atributos, noreturn, por exemplo. Também pode ser incluído especificadores de alinhamento.
- Nome - o identificador da classe. Opcionalmente acompanhado da palavra chave final. Caso o nome seja omitido, a classe é considerada anônima e não pode ser final.
- Classes base - lista opcional de classes pai e o modelo de herança que será usado para cada uma delas.
- Membros - lista de membros, especificadores de acesso, declaração e definição de funções de instância, tipos e classes aninhados.

## 2.5 Classes Abstratas

Classes abstratas são tipos que não podem ser instanciados, mas podem ser usados como classe base. Para criar uma, basta usar o especificador virtual. Classes como essas são usadas para representar conceitos gerais, por exemplo, Forma ou Animal, esses tipos serão usados como classe base para tipos concretos como Círculo ou Cachorro.

Não é possível usar tipos abstratos em parâmetros, tipos de retorno ou conversões. É possível usar esses tipos com ponteiros e referências.

## 2.6 Polimorfismo

Objetos de uma classe que declaram ou herdam pelo menos uma função virtual são considerados polimórficos. Dentro de cada objeto deste tipo, a implementação contém informações adicionais que são usadas por chamadas de funções virtuais e por métodos auxiliares como `dynamic cast` ou `typeid`.

## 2.7 Memória

C++ possui 4 tipos de gerenciamento de memória:

- Armazenamento de duração estática: são criados antes da chamada `main()` (salvo exceções) e destruídos na ordem reversa de criação após a saída de `main()`.
- Armazenamento de duração de thread: Similar ao armazenamento estático, porém o objeto é criado com a thread e destruído com o `join` da thread.
- Armazenamento automático: Variáveis automáticas são criadas no ponto de declaração e destruídas na ordem reversa de criação de seu escopo. São alocadas automaticamente na pilha.
- Armazenamento dinâmico: São criadas com uma chamada `new` e destruídas com uma chamada `delete`.

## 3 IMPLEMENTAÇÃO

### 3.1 Orientação a Objetos

A implementação inicial inclui duas classes bases:

- Jogador: Classe que será utilizada para criar informações sobre o jogador, como nome e pontuação.

Atributos: `std::string nome`, `int pontuacao`.

Métodos: Construtor, `getPontuacao`, `setPontuacao(int pontuacao)`.

- Pergunta: Classe que conterá uma pergunta com suas alternativas e resposta.

Atributos: `std::string enunciado`, `std::string alternativas[4]`, `int resposta`.

Métodos: Construtor, `showPergunta()`: mostra a pergunta e as alternativas, `responder(int alternativa)`: recebe a resposta do usuário, `perguntar()`: realiza a pergunta com os métodos `showPergunta()` e `responder(int alternativa)`.

Além dessas classes, uma classe `Database` está incluída para gravar dados do programa, essencialmente as perguntas, e uma classe `Main` para execução do programa.