# Laboratório III

## Classes, Objetos, Métodos Construtores, Método Destrutor e Sobrecarga de Operadores

## 1. Objetivo

O objetivo deste exercício é colocar em prática conceitos do paradigma de Programação Orientada a Objetos (POO) na linguagem de programação C++, em particular pela implementação de classes, objetos, métodos construtores, método destrutor e sobrecarga de operadores.

## 2. Orientações gerais

Você deverá observar as seguintes observações gerais na implementação deste exercício:

1) Apesar da completa compatibilidade entre as linguagens de programação C e C++, seu código fonte não deverá conter recursos da linguagem C nem ser resultante de mescla entre as duas linguagens, o que é uma má prática de programação. Dessa forma, deverão ser utilizados estritamente recursos da linguagem C++.

2) Você deverá utilizar apenas um editor de texto simples (tais como o Gedit ou o Sublime) e o compilador em linha de comando, por meio do terminal do sistema operacional Linux.

3) Durante a compilação do seu código fonte, você deverá habilitar a exibição de mensagens de aviso (*warnings*), pois elas podem dar indícios de que o programa potencialmente possui problemas em sua implementação que podem se manifestar durante a sua execução.

4) Aplique boas práticas de programação. Codifique o programa de maneira legível (com indentação de código fonte, nomes consistentes, etc.) e documente-o adequadamente na forma de comentários. A título de sugestão, anote o seu código fonte para dar suporte à geração automática de documentação utilizando a ferramenta Doxygen (http://www.doxygen.org/). Consulte o documento extra disponibilizado na Turma Virtual do SIGAA com algumas instruções acerca do padrão de documentação e uso do Doxygen.

5) Busque desenvolver o seu programa com qualidade, garantindo que ele funcione de forma correta e eficiente. Pense também nas possíveis entradas que poderão ser utilizadas para testar apropriadamente o seu programa e trate adequadamente possíveis entradas consideradas inválidas.

6) Para melhor organização do seu código, implemente diferentes funções e faça a separação da implementação do programa entre arquivos cabeçalho (.h) e corpo (.cpp).

## 3. Tarefas

Implemente em C++ as respectivas classes, atributos e métodos (incluindo construtores e destrutor) necessários para atender às seguintes abstrações:

a) Livro possuindo as informações de titulo, autor, editora, ano.

b) Usuário possuindo as informações deNome, endereço, Identificador.

c) Emprestimo que guarda as informações do usuário que realizou o emprestimo, qual livro foi emprestado, data do emprestimo e data de devolução.

Usando as classes implementadas, escreva um programa em C++ para simular uma biblioteca. A biblioteca possui uma coleção de livros onde são disponibilizados para que os usuários possam leva-los por um tempo determinado, por ser uma biblioteca nova e possuir poucos livros, um usuário pode apenas levar um livro por vez.

### 3.1. Funcionalidades

O código desenvolvido deve conter as seguintes funcionalidades:

1. Busca de livros por titulo;
2. Realizar emprestimos;
3. Informar quais emprestimos estão atrasados, mostrando o usuário e o livro emprestado;
4. Inserção de livros e usuários do sistema será pela leitura de arquivos que conterá essas informações.

### 3.2. Organização do Projeto

O código deverá ser devidamente comentado e anotado para dar suporte à geração automática de documentação no formato de páginas Web (HTML) utilizando a ferramenta Doxygen. Para maiores informações, você poderá acessar a página do Doxygen na Internet (http://www.doxygen.org/).

O código do projeto deve seguir a configuração de pastas e arquivos:

1. /bin – código executável
2. /src – código fonte
3. /docs – documentação
4. makefile

5. README – arquivo contendo informações sobre: configuração, compilação e execução. Também deve conter uma sessão com as informações sobre quais arquivos e as linhas que contêm: Uso de alocação dinâmica de memória. Sobrecarga de funções; Pilha e fila implementada para o livro e os usuários; Sobrecarga de Operadores.

## 4. Entrega

Todos os códigos fonte referentes à implementação do trabalho deverão ser disponibilizados **sem erros** de compilação e devidamente testados e documentados através do repositório Git. Além disso, você deverá submeter, até 23:59 do dia 18 de março de 2017, um único arquivo compactado através da opção *Tarefas* na Turma Virtual do SIGAA contendo: (1) os mesmos arquivos de código fonte disponíveis no repositório Git; (2) a documentação do projeto na forma de páginas HTML, geradas automaticamente com a ferramenta Doxygen, e;

É importante destacar que serão avaliados **única** e **exclusivamente** os arquivos submetidos via SIGAA.

## 5. Avaliação

A avaliação deste trabalho será feita principalmente sobre os seguintes critérios: (1) utilização correta dos conteúdos vistos nas aulas presenciais da disciplina; (2) a corretude da execução do programa implementado, que deve apresentar saída em conformidade com a especificação e as entradas de dados fornecidas; (3) a aplicação correta de boas práticas de programação, incluindo legibilidade, organização e documentação de código fonte, e; (4) qualidade do relatório técnico produzido. O trabalho possuirá nota máxima de 4,00 (quatro) pontos para a segunda unidade, distribuídos de acordo com a seguinte composição:

| **Itens Avaliados** | **Nota máxima** |
|---|---|
| Busca de livros por titulo. | 0,50 |
| Realizar emprestimos. | 0,50 |
| Emprestimos atrasados. | 0,70 |
| Inserção de livros e usuários do sistema. | 0,50 |
| Documentação do código. | 0,25 |
| Modularização adequada das Classes. | 0,15 |
| Criação do Makefile. | 0,15 |
| Uso consistente de alocação dinâmica de memória e uso do This. | 0,20 |
| Aplicação adequada de sobrecarga de funções. | 0,25 |
| Uso de listas (Pilha ou fila). | 0,50 |
| Sobrecarga de operadores | 0,30 |
| **Total** | **4** |

Por sua vez, o não cumprimento de algum dos critérios abaixo especificados poderá resultar nos seguintes decréscimos, calculados sobre a nota total obtida até então:

| **Falta** | **Decréscimo** |
|---|---|
| Programa apresenta erros de compilação, não executa ou apresenta saída incorreta | –70% |
| Falta do README | 0,5 |
| Implementação na linguagem C ou resultante de mistura entre as linguagens C e C++ | –30% |
| Programa compila com mensagens de aviso (*warnings*) | –50% |
| Plagiarismo | –100% |