UNIVERSIDADE FEDERAL DE SÃO CARLOS
CAMPUS SÃO CARLOS
JOÃO VITOR BRANDÃO MOREIRA - 407496
LUCAS OLIVEIRA DAVID – 407917
THIAGO FARIA NOGUEIRA - 407534

DOCUMENTAÇÃO JARDIM DE FLORES

SÃO CARLOS – SP
2012

1. INTRODUÇÃO

Neste trabalho, iremos implementar uma variação do jogo "*Flower Garden*", com pequenas mudanças em relação a sua jogabilidade, a fim de torná-lo mais simples e rápido de finalizar.

Inicialmente, se consiste em 4 fundações inicialmente vazias - onde as cartas são armazenadas em ordem crescente, e com naipes iguais -; seis pilhas - contendo quatro cartas aleatórias - e finalmente, um buquê com as 28 cartas restantes.

O objetivo do jogo é colocar as cartas, uma a uma, nas fundações, e uma vez que isso acontece, não é possível retirá-la. Entretanto, simplesmente mover as cartas não é suficiente para alcançar esse objetivo. Para conseguir isso, é necessário combinar as cartas que estão no buquê e nas pilhas a fim de liberar as que estão abaixo das do topo dessa mesma. As pilhas possuem propriedades únicas: só as cartas no topo podem ser movidas, além de só poderem ser colocadas sobre uma carta de número especificamente superior (n +1), sendo o naipe irrelevante. Já o buquê, tem todas suas cartas sempre disponíveis, e uma vez movidas para outra estrutura, não podem retornar. Caso haja uma pilha vazia, é possível colocar qualquer carta nela.

2. ESTRATÉGIA DE IMPLEMENTAÇÃO

O jogo foi implementado na linguagem C++. No inicio, Java foi cogitado, por apresentar facilidades em relação à interface gráfica. Entretanto, pela pouca experiência dos membros do grupo com o mesmo, optamos pelo C++, que foi ministrado no semestre anterior na matéria de Programação de Computadores. A interface gráfica foi desenvolvida utilizando a biblioteca gráfica SDL, e cada componente do grupo utilizou a IDE que mais lhe agradava - Visual Studio e Code::Blocks. Os arquivos que eram compartilhados sempre foram os suportados pelas IDEs citadas (main.cpp, Pilha.h, Carta.cpp, etc), não implicando, assim, em nenhuma incompatibilidade.

A estrutura de pilha foi implementada utilizando template, visando a reusabilidade do código.

O embaralhamento é realizado no inicio do jogo, onde todas as cartas estão no vetor, primeiramente, separadas por naipe, e em seguida, em ordem crescente. Acessando todas as posições do vetor, trocamos a carta na posição atual com a referenciada por um indexador pseudo-aleatório gerado pela função rand() % 52 (considerando que o vetor vai da posição zero até a 51, a operação modular por 52 é indispensável).

A movimentação das cartas é dada pela entrada e saída nas pilhas através de seus métodos básicos (empilha e desempilha).

O buquê foi dividido em duas partes de 14 cartas para não extrapolar o tamanho da tela.

3. DIVISÃO DO TRABALHO

- **João Vitor e Lucas David:** responsáveis pela parte estrutural do jogo, isto é, a implementação dos tipo abstrato de dados utilizado - as pilhas que contém as cartas - e lógica - como o embaralhamento, transição das cartas, regras do jogo, etc; além da documentação.

- **Thiago Faria:** responsável pela interface gráfica do jogo, auxiliar na construção das regras do jogo e a correção de *bugs* referentes à integração entre o trabalho dos componentes do grupo; além da documentação.

4. DESCRIÇÃO DA JOGABILIDADE

4.1 Imagens do jogo em fase de desenvolvimento

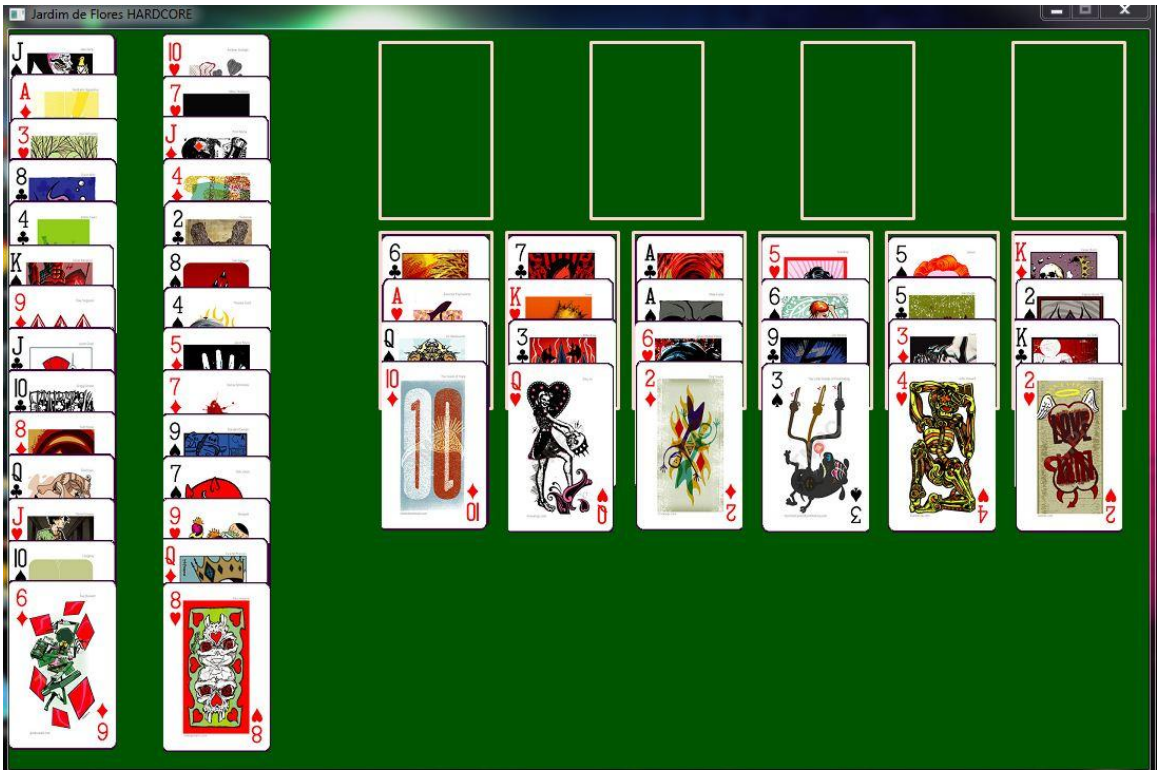

Figura I - formulário principal do jogo v0.5

Para mover uma carta de um lugar pra outro, seja do topo de uma pilha para outra, de uma pilha para uma fundação ou do buquê para pilha ou fundação, é necessário somente clicar sobre a carta desejada e depois clicar onde você deseja colocá-la. Se for possível, a carta sairá da sua origem e irá para o destino.

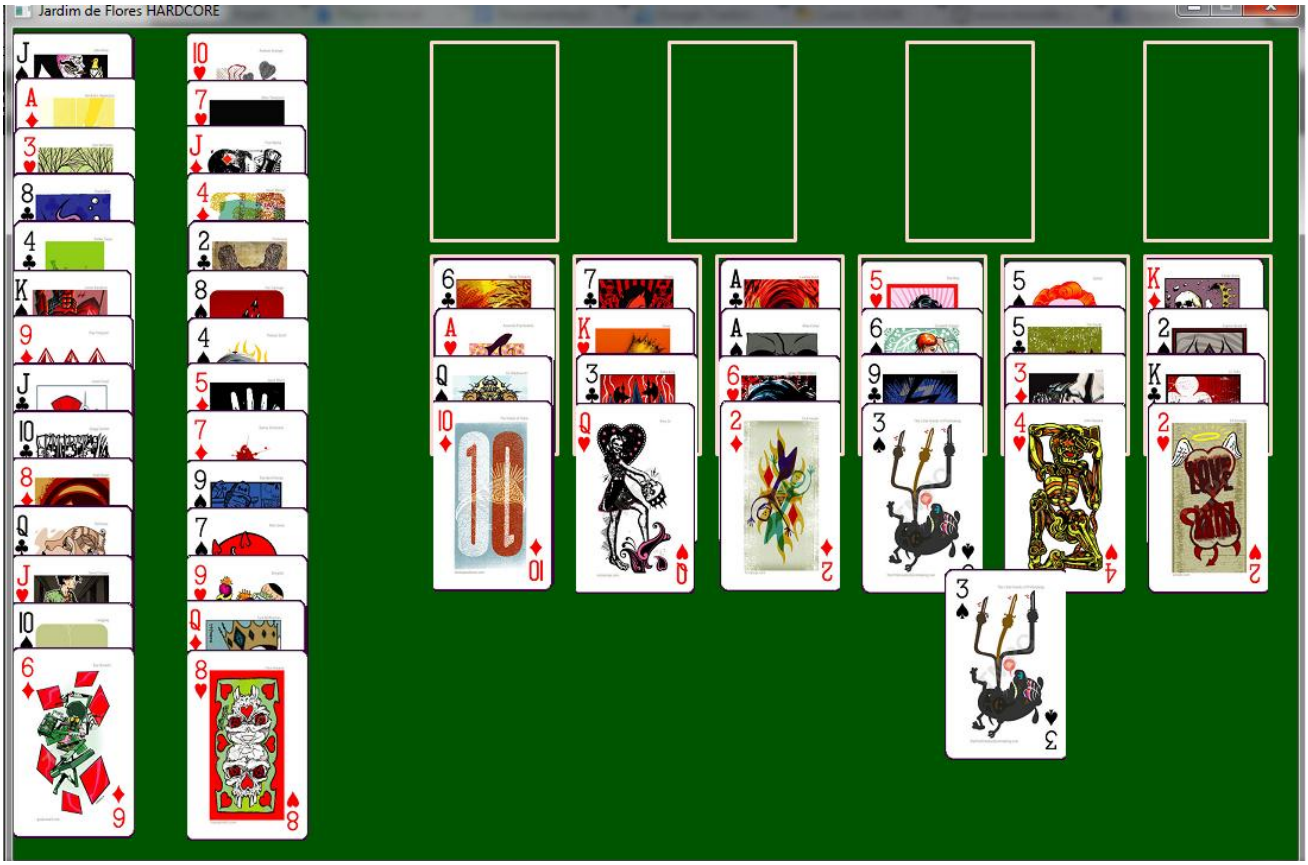

Figura II - movimentação de uma carta da pilha

Perceba que a carta não sai da pilha a não ser que seja possível movê-la, ou seja, se for de uma pilha para outra a carta deve ter número **n -1** em relação do número **n** da carta do topo da pilha onde se deseja mover, e se for para a fundação, deve ter o naipe igual e o número **n +1** em relação ao número n da carta do topo da fundação.

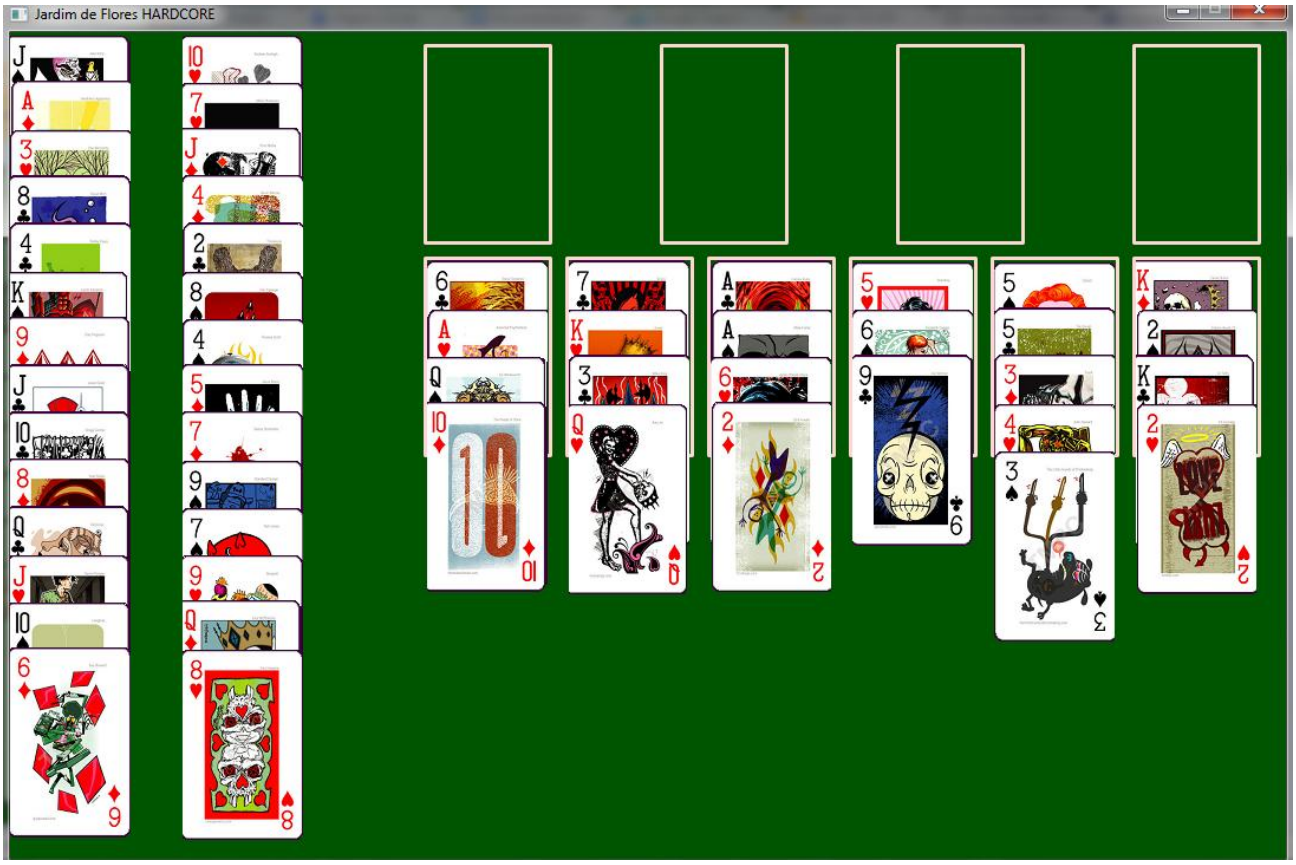

Figura III: sobreposição da carta 4 pela carta 3, movida da a pilha à esquerda

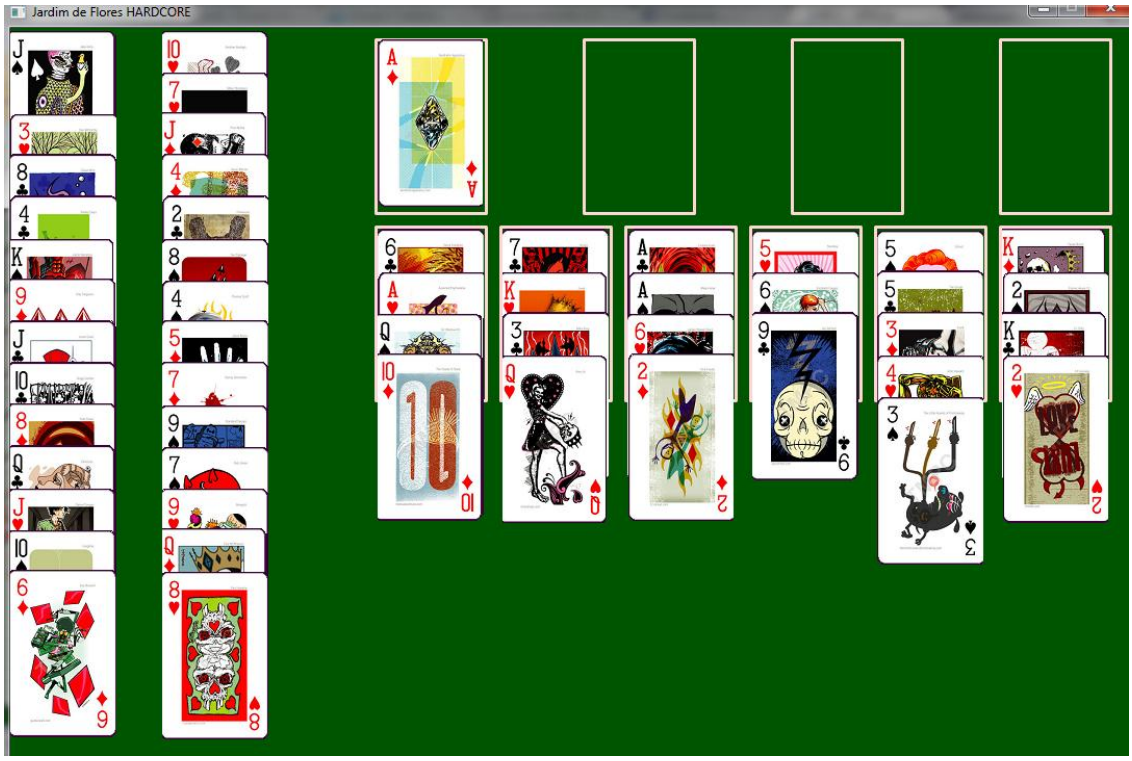

Figura IV: movimentação do Ás de ouros do buquê para uma fundação

Figura V: Movimentação do 2 de ouros da terceira pilha para a primeira fundação

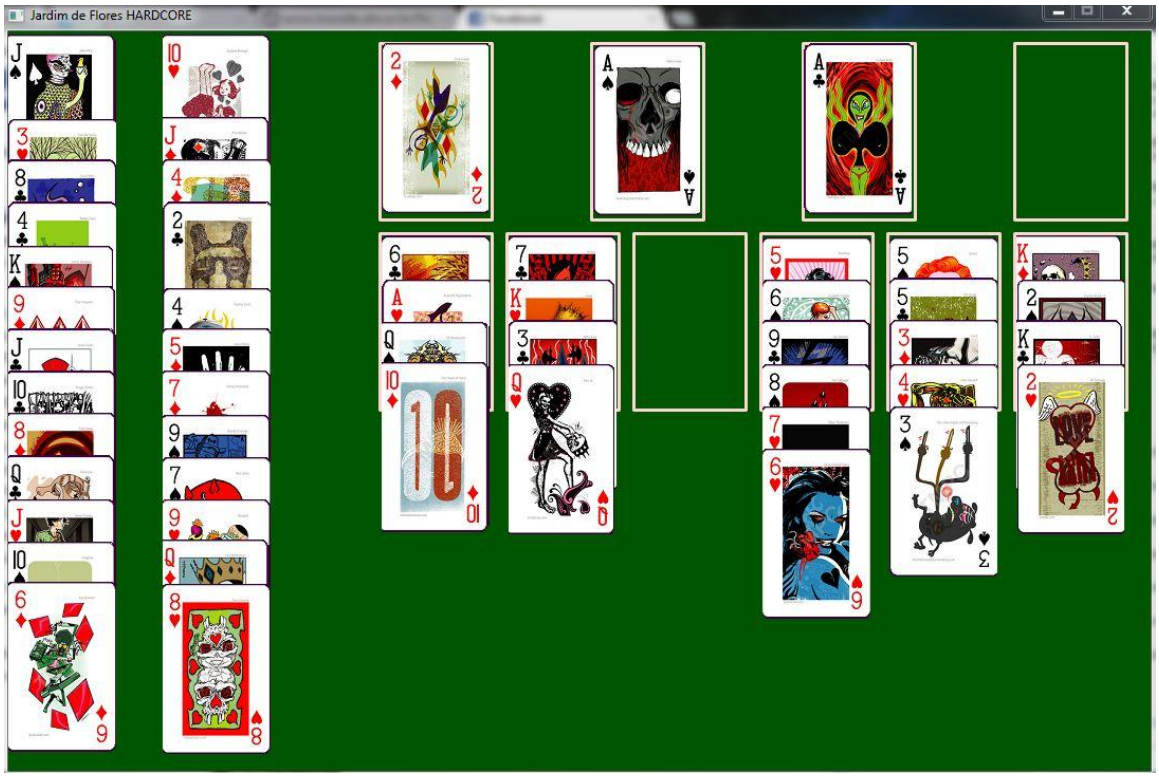

Figura VI: resultado de sucessivas ocorrências para esse mesmo jogo

4.2 Imagens do jogo em fase de teste

Figura VII: um caso em que uma pilha fica vazia

Figura VIII: moveu-se uma carta qualquer do buquê para a pilha anteriormente vazia

4.3 Imagens do jogo em fase de distribuição

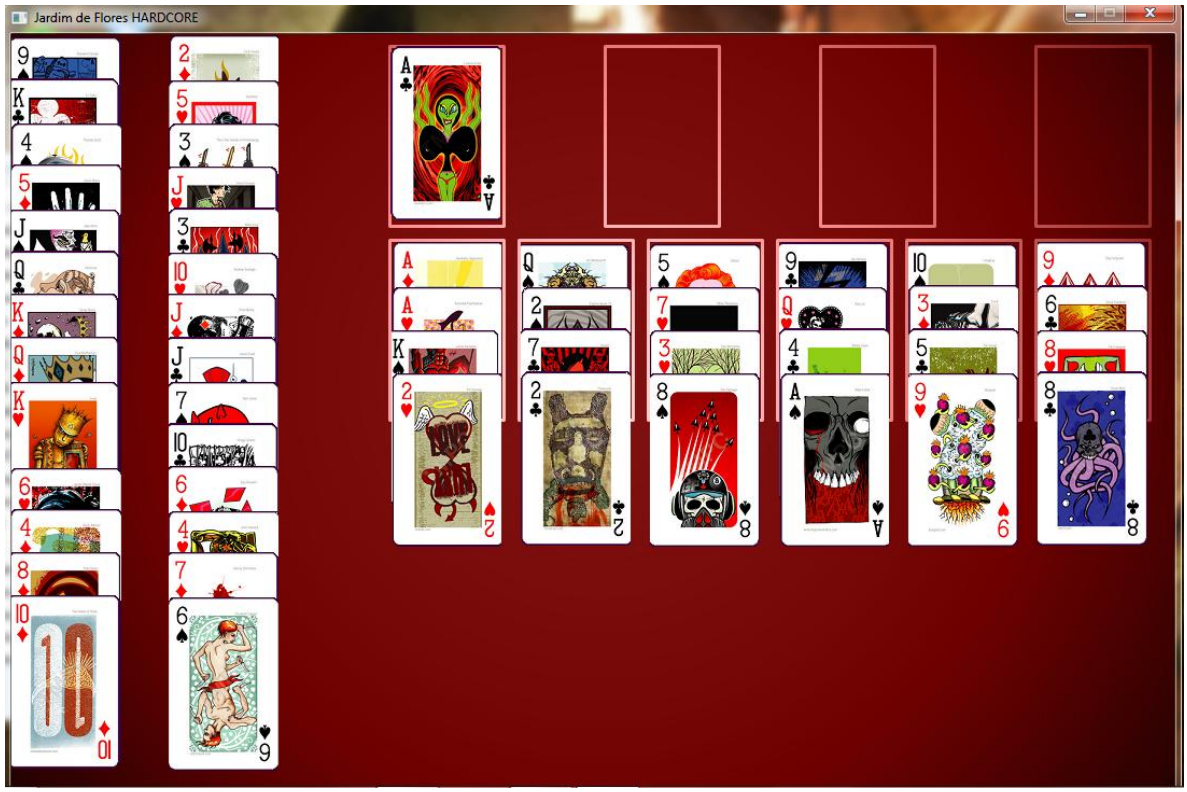

Figura IX - formulário principal do jogo v1.0

CONCLUSÃO

O jogo, apesar de possuir regras mais simples, em relação ao FreeCell, ofereceu certas barreiras durante sua implementação, devido à utilização de recursos como o polimorfismo paramétrico (template) e interface gráfica. Entretanto, percebemos a importância de tal implementação: as facilidades em se alocar diferentes tipos de dados às pilhas: possibilitará que este mesmo código seja reutilizado em projetos futuros na matéria de Estrutura de Dados ou suas subsequentes. A interface gráfica, por sua vez, oferece ao jogo um ambiente muito mais atraente, convidativo e intuitivo aos usuários, essencial para tornar o jogo divertido; a biblioteca SDL atendeu bem às nossas expectativas iniciais, porporcionando gráficos bem feitos e sendo de fácil manuseio. No início, o método de embaralhamento era mais complexo, demandando mais desempenho do computador - cada carta, aleatóriamente, era enviada para um dos seis vetores existentes. Ao final, todos eram reunidos novamente, e o processo era repetido, para promover uma melhor mistura de naipes e números. O desenvolvimento de métodos básicos da pilha tornou o desenvolvimento do jogo mais fácil, uma vez que, após concluído, pudemos nos concentrar em desenvolver as regras do jogo, além de abrir a possibilidade de também ser utilizado em implementações futuras. Por fim, observamos a dificuldade de implementação que a linguagem C++ impõe, em relação ao JAVA e ao C#.