Universidade Estadual de Santa Cruz
Departamento de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação
Disciplina: Linguagem de Programação II – 2013.1

Especificação de Projeto de Programação

Jogo da Velha

Professor

Paulo Costa

Ilhéus, BA
Março de 2013

Sumário

Lista de Tabelas

Lista de Figuras

1. Objetivo

O objetivo geral deste projeto é aplicar princípios e técnicas de programação, como *funções*, *encapsulamento de dados*, *modularização de código* e *compilação em separado*, ao jogo da velha. O objetivo específico do projeto é implementar as dez funções descritas a seguir.

2. Descrição

O ponto de partida do projeto é uma implementação completa do jogo da velha. O programa, escrito em C, encapsula as estruturas de dados e suas funções de acesso, simulando objetos e métodos encontrados em linguagens orientadas a objetos.

O programa foi dividido em seis módulos, correspondendo aos seis arquivos `.c` do projeto. Após testado, o código das dez funções listadas na Tabela 1 foi removido. Sua tarefa é re-implementá-las de modo que o programa execute corretamente.

Tabela 1 – Funções a serem implementadas.

Módulo	Função	Objetivo
`objetos`	`lerVitoriasJogador`	Obter o total de vitórias de um jogador
	`lerDerrotasJogador`	Obter o total de derrotas de um jogador
`interface`	`mostrarTabuleiro`	Mostrar na tela o tabuleiro com as peças jogadas até então e uma legenda com a numeração das casas
	`obterSIMouNAO`	Pedir ao usuário que responda sim ou não a uma pergunta e retornar a resposta
`estrategia`	`casasAlinhadas`	Determinar se duas casas no tabuleiro estão numa linha horizontal, vertical ou diagonal
	`terceiraCasaDaLinha`	Se duas casas no tabuleiro estiverem numa linha, determinar o nº da terceira casa nessa linha
	`casaParaVitoria`	Determinar se certo jogador pode ganhar jogando em alguma casa
	`proximaJogadaPrograma`	Determinar a casa onde o programa deve fazer sua próxima jogada
	`venceuPartida`	Determinar se certo jogador venceu a partida
`torneio`	`realizarPartida`	Conduzir uma partida até a vitória ou empate

3. Realização

O trabalho deve ser realizado em duplas. Se o tamanho da turma for ímpar, o aluno que não se encaixar em nenhum grupo deve procurar o professor da disciplina pelo menos duas semanas antes da data da entrega. Em nenhuma outra circunstância será aceito trabalho que não em dupla.

3.1. Desenvolvimento

Há três opções de ambiente para desenvolvimento para a realização do projeto:

- Em Windows, utilizando Dev-C++.
- Em Windows, utilizando CodeBlocks.
- Em Linux, utilizando qualquer editor de texto e gcc.

As duas IDEs de Windows (Dev-C++ e CodeBlocks) são gratuitas e facilmente encontradas na internet. A Tabela 2 mostra as versões atuais de cada uma delas, bem como os *links* para sua obtenção. Escolha e use apenas um dos três ambientes para realizar o trabalho e ignore os demais.

Tabela 2 – URLs para obtenção das IDEs

Code::Blocks para Linux, Windows ou MacOS: `http://www.codeblocks.org/downloads/26`
Dev-C++ 4.9.9.2 para Windows: `http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe`

Observe que o código está distribuído em diversos arquivos, mas você só deverá modificar e entregar os quatro arquivos listados na Tabela 1, ou seja:

```
objetos.c
estrategia.c
interface.c
torneio.c
```

Modifique somente as funções listadas na Tabela 1 ***e nada mais***. Após incluir seu código nesses arquivos, compile o programa. Se estiver usando a linha de comando de Linux, digite `make clean` e depois `make all`. Se o comando `make` não estiver instalado no seu Linux, digite `./Compilar` .

Qualquer que seja o ambiente de desenvolvimento escolhido para realizar o trabalho, a compilação cria o executável `velha` em Linux ou `velha.exe` em Windows, que deve ser invocado sem argumentos. Execute-o, compare seu funcionamento com o do programa modelo (veja seção 3.2 a seguir) e ajuste seu código conforme necessário.

3.2. Funcionamento

A Figura 1 (p. 3) ilustra o funcionamento esperado do programa. Esse diagrama não mostra detalhes da interação do programa com o usuário. Para isso, utilize como modelo o executável fornecido pelo professor (`velha-windows.exe` ou `velha-linux`). Execute-o com todas as combinações de entradas, inclusive entradas inválidas. O termo *entrada*, no contexto deste projeto, significa tudo que é fornecido pelo teclado pelo usuário em resposta às perguntas feitas pelo programa, por exemplo

```
Digite seu nome:
Joga em qual casa?
Deseja jogar outra partida [S/N] ?
```

Observe atentamente o conteúdo e formatação de tudo que é mostrado na tela, inclusive mensagens de erro. Sua implementação deverá funcionar de maneira exatamente idêntica ao programa modelo, incluindo a interação com o usuário. Qualquer desvio ou inconsistência, por menor que seja, resultará em desconto na nota. Maiores detalhes sobre a avaliação do trabalho na seção 6 (p. 4).

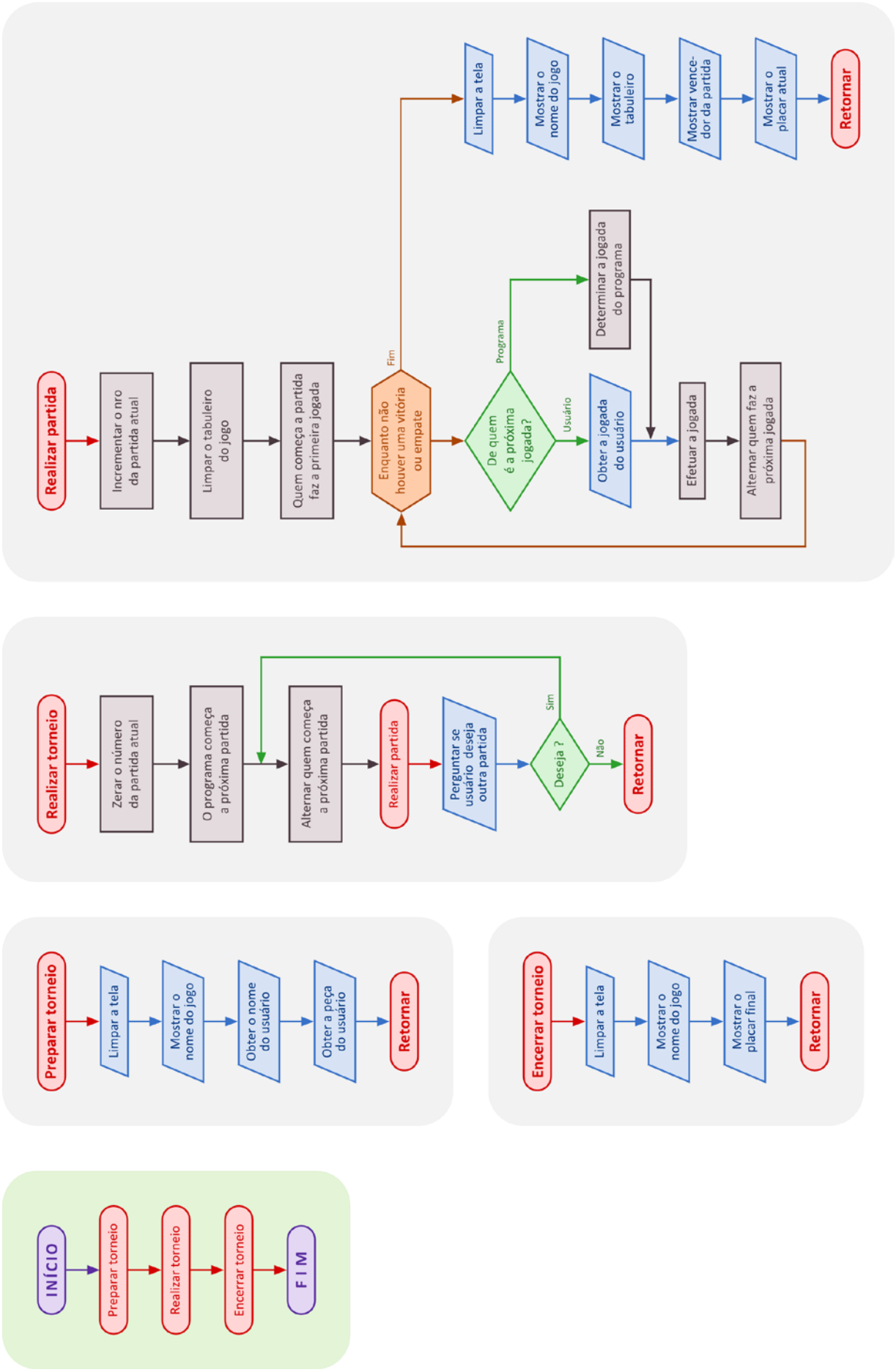

Figura 1 – Funcionamento esperado do programa.

4. Teste

É fundamental testar seu código com várias entradas diferentes, válidas e inválidas, em todas as combinações, até se certificar que está totalmente correto. Execute o programa fornecido pelo professor (`velha-windows.exe` ou `velha-linux`) e compare o funcionamento com o do seu código. Ambos deverão funcionar de maneira exatamente idêntica, inclusive as mensagens impressas na tela (conteúdo e formatação, até mesmo os espaços em branco).

5. Entrega

Quando terminar o trabalho, crie uma pasta com os nomes dos autores, p.ex.

```
JoseCarlos-AnaMaria
```

Use apenas iniciais de cada nome maiúsculas, demais letras minúsculas e hífen. ***Não use*** números, letras acentuadas, espaços em branco, sublinhados, caracteres especiais, etc. Coloque nessa pasta os quatro arquivos contendo as funções que você implementou. Compacte a pasta criando um arquivo `.zip`, p.ex.

```
JoseCarlos-AnaMaria.zip
```

Use qualquer ferramenta de compactação disponível, mas certifique-se que o arquivo compactado é do tipo `zip` e não qualquer outro como `rar`, `tar`, `7z`, `tgz`, etc. Envie esse arquivo por email para

```
paulocostauesc@yahoo.com.br
```

até as 23h59m do dia da entrega. O assunto do email deve ser

```
Projeto de LP2 jogo da velha
```

O corpo do email deve conter os números de matrícula e os nomes completos dos autores, p.ex.

```
201000123 Jose Carlos Pereira
201000456 Ana Maria Braga
```

Não esqueça de anexar ao email o arquivo zipado para avaliação. Se um trabalho for enviado mais de uma vez, apenas a versão enviada por último será corrigida; todas as anteriores serão descartadas. Em hipótese alguma serão admitidas correções, substituições ou ajustes de qualquer natureza após o prazo de entrega.

6. Avaliação

6.1. Aceitação do trabalho

Não será aceito nenhum trabalho

- realizado individualmente ou em grupo que não de 2 alunos, exceto com autorização prévia;
- implementado em qualquer linguagem que não C;
- contendo qualquer modificação no código além do especificado neste documento;
- entregue por qualquer meio que não o especificado neste documento;
- entregue após o prazo.

Em qualquer destes casos a nota será zero.

6.2. Compilação do código

O código entregue será integrado ao restante do código fonte usado para teste dos trabalhos. A seguir, será compilado com `gcc` e as *flags* `-Wall –pedantic`, as mesmas usadas nos projetos implementados e disponibilizados pelo professor.

- Se o compilador não gerar erros ou *warnings*, o trabalho será corrigido normalmente.
- Se o compilador não gerar erros mas gerar *warnings*, o trabalho será corrigido normalmente, mas haverá desconto na nota. *Warnings* não impedem a geração de código executável, nem implicam que o código executável produzirá erros de execução ou resultados incorretos.
- Se o compilador gerar erros, o trabalho não será corrigido e a nota será zero. Erros de compilação impedem a geração de código executável, impossibilitando o teste do programa.

6.3. Execução do programa

Uma vez compilado, o programa será testado extensivamente, com diferentes entradas. Os mesmos parâmetros de execução serão aplicados igualmente a todos os trabalhos, de modo a uniformizar as avaliações.

Se um programa produzir qualquer erro de execução, a nota no trabalho será zero. Erros de execução são aqueles que causam o término anormal ou prematuro do programa, geralmente como consequência de uma violação de acesso a memória. Essa definição não diz respeito à qualidade ou correção dos resultados produzidos pelo programa, mas à normalidade de sua execução e término. Assim, um programa que produz resultados incorretos será corrigido, desde que seu término seja normal, ou seja, ocorra no tempo certo.

6.4. Pontuação do trabalho

Sua nota será 10 se:

- os nomes de variáveis forem bem escolhidos,
- os comentários forem suficientes, informativos e linguisticamente corretos,
- os algoritmos forem bem implementados,
- o programa compilar e executar corretamente, e
- seu funcionamento for idêntico ao do programa modelo.

Caso contrário, descontos serão aplicados à nota, conforme mostrado na Tabela 3.

6.5. Casos omissos

A avaliação do trabalho será guiada pelos critérios aqui descritos, mas não será limitada a eles. Poderá haver descontos na nota em consequência de situações que, embora não previstas neste documento, demonstrem uma compreensão ou execução incorreta ou imperfeita do trabalho. Quando isso ocorrer, os descontos na nota serão devidamente justificados.

Tabela 3 – Descontos a serem aplicados à nota.

	Peso	Critério	Cód	Ocorrência	Desconto	
					Total	Cada
Compil.	10	Compilação e execução	01	Compilação gera erro	10,0	
	10		02	Compilação gera warning		1,0
	10		03	Execução gera erro	10,0	
Qualidade do código	6	Variáveis	04	Nomes pouco informativos		2,0
	6		05	Nomes curtos demais ou longos demais		2,0
	6		06	Variáveis desnecessárias		2,0
	6	Comentários	07	Comentários irrelevantes ou pouco informativos		2,0
	6		08	Comentários insuficientes	10,0	
	4		09	Linguagem inapropriada ou erros de português	10,0	
	4	Algoritmos	10	Limitações arbitrárias ou incorretas (tamanho arrays etc.)	0,0	
	4		11	Lógica confusa, complicada ou deselegante	10,0	
	4		12	Código condensado demais	10,0	
Comportamento	8	Reação a entradas inválidas	13	Escolha da peça: resposta diferente de 1 e 2 para X e O	10,0	
	8		14	Casa em que deseja jogar é menor que 1	10,0	
	8		15	Casa em que deseja jogar é maior que 9	10,0	
	8		16	Casa em que deseja jogar está ocupada	10,0	
	8		17	Deseja jogar outra partida: resposta diferente de S e N	10,0	
	8	Partidas e torneios	18	Jogada do programa diferente da jogada do modelo	10,0	
	8		19	Detecção incorreta de vitória	10,0	
	8		20	Detecção incorreta de empate	10,0	
	8		21	Mensagens na tela com conteúdo incorreto		3,0
	8		22	Mensagens na tela com formatação diferente do modelo		3,0