Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Trabalho Prático 01 (T1)

Identificação da Disciplina

Código da Disciplina	Nome da Disciplina	Turma	Professor	Período
113476	Algoritmos e Programação de Computadores	H	Jeremias Moreira Gomes	2019/2

1. Objetivo

1.1. Objetivo

O objetivo deste trabalho é desenvolver um jogo interativo com um usuário, por meio do terminal.

Ao cumprir os objetivos do trabalho (T1), o aluno deverá ter: (*i*) exercitado suas habilidades de programação contemplando estruturas condicionais, estruturas de repetição e estruturas homogêneas; (*ii*) dado o enunciado, ter resolvido um problema, elaborando um algoritmo para solucioná-lo; e (*iii*) desenvolver um jogo de RPG que funcione corretamente em um computador.

2. Enunciado

A empresa Universo Brasileiro de Games (UnBG) contratou você para revolucionar o mundos dos jogos, por meio da conquista nostálgica da população dos dias atuais. A UnBG desenvolve jogos do tipo *Role Playing Game* (RPG) e, para testar seus conhecimentos e também conquistar o público *gamer*, decidiu que você irá desenvolver um RPG de texto do tipo Caverna Multiusuário (MUD (*Multi-user dungeon*)).

Em termos de jogos de computador, um MUD é um RPG multijogadores, que normalmente é executado em um servidor na Internet. Esses servidores na internet costumam utilizar um protocolo chamado Telnet, que permite aos jogadores acessarem o jogo por meio de terminais.

Já em RPG, os jogadores assumem o papel de um personagem e recebem informações textuais que descrevem salas, objetos, outras personagens e criaturas controladas pelo computador, também conhecidas como *non-player characters* (NPCs), em um mundo virtual. Eles podem interagir com outros jogadores e personagens digitando comandos que, normalmente, se encontram em inglês.

Um RPG de texto consiste em um jogo onde o jogador deverá criar seu personagem fictício e receber uma história com eventos. A cada evento, uma decisão deve ser tomada e um novo pedaço da história deve ser contado. Dependendo das escolhas realizadas pelo jogador, ele poderá concluir a história com um desfecho diferente.

Um exemplo desses jogos é O Mundo de Achaea (`https://www.achaea.com/`), e para acessar esse jogo, como exemplo, você pode digitar o seguinte comando no seu terminal (se você tiver o protocolo telnet nele, que á padrão na maioria dos sistemas operacionais): `telnet achaea.com`.

Um exemplo de saída para esse comando é a seguinte:

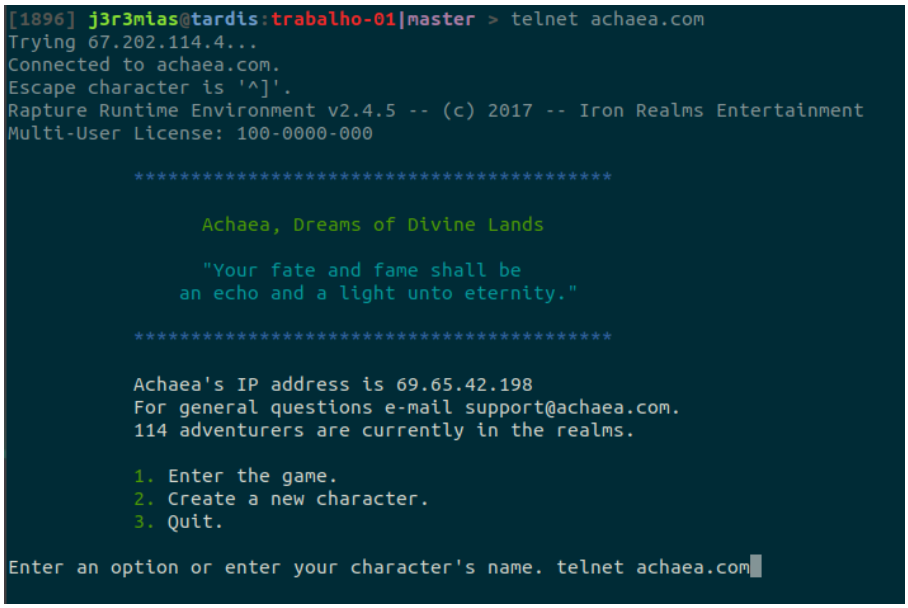

Figure 1. Exemplo da Tela Inicial do Jogo O Mundo de Achaea.

3. Metodologia

O jogo deverá ser desenvolvido em quatro partes: (*i*) Criação de um menu de apresentação; (*ii*) Criação de personagem; (*iii*) Apresentação da narrativa de uma história; e (*iv*) Criação de subjogos.

3.1. Menu de entrada (10 pts)

Crie um breve diálogo mostrando 3 opções ao usuário.

1. Criar Personagem
2. Iniciar Jogo
3. Sair

O usuário só poderá sair do jogo, se escolher a opção de número 3. Caso contrário, ele terá acesso a parte de criação de personagem (opção 1) ou começará o jogo (opção 2). A opção de iniciar jogo só poderá ser acessada, caso a primeira opção já tenha sido utilizada. A primeira opção poderá ser acessada quantas vezes o jogador quiser para se alterar as características do personagem antes do jogo começar.

3.2. Criação de Personagem (30pts)

O jogador deverá ser capaz de criar seu personagem com as seguintes características:

- Nome - Nome do seu personagem (texto com até 20 caracteres entre letras e números somente).
- Raça (1 - Humano, 2 - Anão, 3 - Elfo)
- Alinhamento (1 - Mal, 2 - Neutro e 3 - Bom)

- Profissão (1 - Guerreiro, 2 - Mago, 3 - Ladino)
- Meta (1 - Governar o reino, 2 - Ficar rico, 3 - Destruir o mal)
- Meio em que vive (1 - Urbano, 2 - Rural, 3 - Tribal)
- História Prévia - Como o personagem viveu até aquele momento (Texto de 400 caracteres no máximo)
- Porte (1 - Grande, 2 - Médio, 3 - Pequeno)

***Restrições de Opções**

1. Quando um personagem for da raça anão, ele só poderá escolher o porte como médio ou pequeno.
2. Quando um personagem for de alinhamento Bom, ele não poderá escolher a classe Ladino.
3. Se a classe Ladino for escolhida, o meio em que o personagem vive não pode ser tribal.
4. Se o alinhamento do personagem for Mal, ele não poderá ter a meta destruir o mal.

Além disso, o personagem irá iniciar com 100 pontos de vida e se ao longo do jogo, os pontos de vida chegarem a 0, o jogo irá terminar com *Game Over*.

3.3. Apresentação da narrativa de uma história (30 pts)

O jogo deverá apresentar uma história recheada de eventos. Você deverá permitir que essa história seja contada ao jogador. Os eventos, quando apresentados, serão menus enumerados sempre com três opções (1, 2, 3). Serão, no total, 5 eventos.

Condução:

1. Crie uma pequena introdução de um mundo fantástico e conte ao jogador onde ele se encontra.
2. Crie um evento (1) onde o jogador poderá escolher entre 3 opções enumeradas.
3. A partir da decisão do jogador, insira uma pequena narrativa.
4. Crie um evento (2) onde o alinhamento do jogador eliminará alguma possibilidade.
5. A partir da decisão do jogador, insira uma pequena narrativa.
6. Crie um novo evento (3) onde existirá uma opção que fará o jogador voltar ao passo 4 e outra opção que irá se basear na profissão do personagem para ser apresentada (Por exemplo, Ladino tem a opção 3 alterada para “Dar uma cambalhota e escapar dos goblins”; enquanto o guerreiro tem a opção 3 alterada para “Empunhar a espada e batalhar com os goblins”).
7. A partir da decisão do jogador, insira uma pequena narrativa.
8. Crie um próximo evento (4), onde o jogador joga jokenpô contra a maquina para poder acessar o último evento.
9. Crie o último evento (5), que será o evento clímax, onde o personagem batalha com uma entidade poderosíssima (invente uma!) que apresentará enigmas e o jogador deverá acertar as respostas de algumas charadas.
10. Caso as charadas sejam resolvidas, apresente o final da história com uma narrativa, caso contrário *Game Over*.

3.4. Criação de Subjogos (30 pts)

Durante a criação do jogo principal, haverão sub-desafios. Esses desafios se resumirão em dois jogos, (*i*) Jokenpô e (*ii*) Três charadas.

3.4.1. Jokenpô (15pts)

O jogador irá jogar jokenpô contra o computador e disputar algumas partidas. Caso o jogador perca, ele deverá perder 10 pontos de vida e, para passar deste desafio, ele terá que alcançar duas vitórias. O jogo de jokenpô deverá ser implementado utilizando uma de três opções (1-pedra, 2- papel e 3-tesoura). A jogada do computador deverá ser executada através de um número pseudo-aleatório entre 1 e 3.

Para utilizar números pseudo-aleatórios em C, você deverá utilizar as bibliotecas `<stdlib.h>` e `<time.h>`. Assim no início da função *main*, o seguinte trecho de código deverá ser utilizado:

```
srand(time(NULL));
```

Isto irá inicializar o gerador de números pseudo-aleatórios a partir da hora atual do computador e para sortear um número entre 0 e 10, por exemplo, você poderá utilizar:

```
x = rand() % 11;
```

3.4.2. Três Charadas (15 pts)

O jogador deverá acertar a resposta de três charadas. Para cada charada, o jogador deve receber um pequeno texto contendo três opções (1, 2 e 3). Das três opções, apenas uma será a resposta da charada. Se o jogador errar a resposta, ele deverá perder 50 pontos de vida. As charadas deverão ser implementadas e aleatoriamente sorteadas a partir de um conjunto de dez charadas. Além disso, as 3 charadas apresentadas ao jogador não poderão ser repetidas entre si.

4. Detalhes Acerca do Trabalho

4.1. Restrições deste Trabalho

Esta atividade pode ser feita em dupla.

4.2. Datas Importantes

Esse trabalho poderá ser submetido a partir de 28/10/2019 (terça-feira) - 00:00h até 03/11/2019 (domingo) - 23:55h.

4.3. Por onde será a entrega o Trabalho?

O trabalho deverá ser entregue via Moodle (https://moodle.cic.unb.br), na disciplina de Algoritmos e Programação de Computadores. Na disciplina haverá uma atividade chamada “Trabalho Prático 01 (T1)”, para submissão do trabalho.

4.4. O que deverá ser entregue, referente ao trabalho?

Deverá ser entregue o código produzido pelos alunos, relacionado a resolução do trabalho.

4.5. Como entregar o trabalho?

A submissão das atividades deverá ser feita em um arquivo único comprimido do tipo zip (`Zip archive data, at least v1.0 to extract`), contendo um diretório com o código elaborado. Além disso, para garantir a integridade do conteúdo entregue, o nome do arquivo comprimido deverá possuir duas informações (além da extensão `.zip`):

- As matrículas dos alunos, separada por hífen.
- O *hash* md5 do arquivo e `.zip`.

  Exemplo: `160068000-180042000-3b1b4a8e9a29cd6f91417d044e67fff2.zip`

Para gerar o md5 do arquivo comprimido, utilize o comando `md5sum` do Linux e em seguida faça o renomeamento utilizando o *hash* coletado.

Segue um exemplo de geração do arquivo final a ser submetido no moodle, para um aluno:

```
[13930] j3r3mias@tardis:trabalho-01 > ls
trabalho-01-apc-jeremias.c
[13931] j3r3mias@tardis:trabalho-01 > zip -r aaa.zip trabalho-01-apc-jeremias.c
  adding: trabalho-01-apc-jeremias.c (deflated 99%)
[13932] j3r3mias@tardis:trabalho-01 > ls
aaa.zip  trabalho-01-apc-jeremias.c
[13933] j3r3mias@tardis:trabalho-01 > ls -lha aaa.zip
-rw-rw-r-- 1 j3r3mias j3r3mias 335 out 15 16:54 aaa.zip
[13934] j3r3mias@tardis:trabalho-01 > md5sum aaa.zip
44cf46f9237cb506ebde3e9e1cd044f9  aaa.zip
[13935] j3r3mias@tardis:trabalho-01 > mv aaa.zip 160068000-44cf46f9237cb506ebde3e9e1cd044f9.zip
[13936] j3r3mias@tardis:trabalho-01 > ls -lha 160068000-44cf46f9237cb506ebde3e9e1cd044f9.zip
-rw-rw-r-- 1 j3r3mias j3r3mias 335 out 15 16:54 160068000-44cf46f9237cb506ebde3e9e1cd044f9.zip
[13937] j3r3mias@tardis:trabalho-01 > md5sum 160068000-44cf46f9237cb506ebde3e9e1cd044f9.zip
44cf46f9237cb506ebde3e9e1cd044f9  160068000-44cf46f9237cb506ebde3e9e1cd044f9.zip
[13938] j3r3mias@tardis:trabalho-01 >
```

4.6. Quem deverá entregar o trabalho?

Ambos os alunos da dupla deverão submeter o trabalho no Moodle. Para isso, **somente um aluno irá gerar a versão final do trabalho (arquivo zip)**, para que o *hash* do arquivo não corra o risco de ficar diferente.

4.7. Observações importantes

- Não deixe para fazer o trabalho de última hora.
- Os trabalhos serão corrigidos pelos monitores.
- Não deixe para fazer o trabalho de última hora.
- Falta de indentação nos programas e ausência de comentários (cabeçalho com descrição sumária e explicação ao longo do código) implicará em redução de pontos.
- Não deixe para fazer o trabalho de última hora.
- Não serão aceitos trabalhos com atraso.
- Cópias de trabalho receberão zero (além disso, plágio é crime).
- Não serão aceitos trabalhos com atraso.
- Não deixe para fazer o trabalho de última hora.
- Os recursos disponíveis no LINF são limitados.
- Não deixe para fazer o trabalho de última hora.
- Programas poderão ser feitos fora do LINF, porém utilizando a linguagem C ANSI.
- **Não deixe para fazer o trabalho de última hora**.

4.8. Utilização do LINF

Para a utilização do LINF, o aluno deve concordar em seguir as regras que regem o seu funcionamento, responsabilizando-se pelos equipamentos que estiver usando. É proibida a ingestão de alimentos nas dependências do LINF. Quando o aluno sair do Laboratório, deverá encerrar a sua sessão e desligar o micro, para evitar que outras pessoas utilizem indevidamente o equipamento, em sessão que está aberta no nome deste aluno. É expressamente proibido o uso dos equipamentos do LINF para jogar qualquer tipo de jogo de computador, exceto para fins de teste quando o jogo estiver sendo desenvolvido no âmbito da disciplina.

5. Bibliografia

1. Cormen, T. et al., Algoritmos: Teoria e Prática. 3a ed., Elsevier - Campus, Rio de Janeiro, 2012
2. Ziviani, N., Projeto de Algoritmos com implementação em Pascal e C, 3a ed., Cengage Learning, 2010.
3. Felleisen, M. et al., How to design programs: an introduction to computing and programming, MIT Press, EUA, 2001.
4. Evans, D., Introduction to Computing: explorations in Language, Logic, and Machines, CreatSpace, 2011.
5. Harel, D., Algorithmics: the spirit of computing, 3a ed., Addison-Wesley, 2004.
6. Marcos A, Furlan et al., Algoritmos e Lógica de Programação., Cengage, 2011.
7. Kernighan, Brian W; Ritchie, Dennis M., C - a linguagem de programacao: Padrao ansi. Ed. Campus, 1989.
8. Farrer, Harry et al., Algoritmos estruturados. 3a ed., Editora LTC, 2011.