

**Departamento de Ciência da Computação  
Métodos de Programação 1/2014**

**Projeto de Disciplina**

**Data de Entrega Especificação: 10/Junho/2014 ate as 23:55**

**Data de Entrega Projeto: 1/Julho/2014 ate as 23:55**

**Jogo de Estratégia**

**Descrição**

Deve ser feito um jogo de estratégia utilizando a interface ncurses (mais detalhes abaixo).

Interface – menu :

O jogo deve ter um menu com pelo menos as opções novo jogo, salvar, carregar o jogo e sair.

Especificação:

No jogo o jogador enfrenta a CPU em um jogo de estratégia. Existe um mapa que mostra todas as informações relevantes.

**A temática e elementos do jogo são de livre escolha desde que não tenham conteúdo que possa ser considerado ofensivo.**

Devem existir pelo menos dois tipos de recursos necessários para criar as unidades (energia e material por exemplo). A criação de cada unidade requer uma certa quantidade destes recursos. Com o recurso igual a zero é impossível criar novas unidades. Estes recursos devem ser renovados de alguma forma. Eles podem ser encontrados no mapa ou gerados através de alguma unidade (ex. construir uma usina que gera energia).

Deve ser possível construir pelo menos 3 tipos de prédios. Estes produzem as unidades de combate com uma certa velocidade.

Devem existir pelo menos 3 tipos de unidades de combate. Elas devem ter forças e fraquezas. Por exemplo, a unidade A pode ser forte contra a unidade B e fraca contra a unidade C.

As unidades podem ser movimentadas no mapa e podem destruir outras unidades ou prédios (dependendo das forças e fraquezas). As unidades podem eventualmente ser destruídas. Elas devem ter característica como velocidade, resistência a danos, etc.

O jogo deve ter um critério de finalização como, por exemplo, se passou um tempo determinado, se todas as unidades e prédios foram destruídas, etc.

O jogo pode ser em tempo real ou baseado em turnos.

A CPU deve ter alguma estratégia para ganhar o jogo.

Um documento descrevendo a especificação do jogo e os integrantes do grupo deve ser enviada até **10/Junho/2014 as 23:55**

**Envios após esta data terão desconto na nota final do trabalho de 2 pontos.**

O documento teve conter os integrantes do grupo e uma descrição de qual será o tema do jogo, prédios, unidades, funcionamento, critério de fim de jogo, etc.

1) Devem ser aplicado neste trabalho todos os conceitos vistos nos trabalhos anteriores.

a) Modularização (makefile, .h e .c),

b) Testes CUnit

c) Assertivas de entrada e de saída. Estas assertivas devem ser colocadas no código através do comando **assert** ou **ifs**. Comentários no código também devem especificar quais são estas assertivas

d) Para cada uma das funções adicionar comentários indicando qual são as interfaces explícita, implícita com a devida descrição, quais são os requisitos e

as hipóteses de cada função. Além disto as funções devem ter finalização adequada liberando memória, fechando arquivos, etc.

e) Faça a modelagem física das estruturas de dados. Use cabeças de estrutura de dados.

f) Assertivas de entrada e de saída como parte da especificação (comentários antes das funções).

g) Assertivas como comentários de argumentação.

h) Assertivas estruturais: elas definem a validade de uma coletânea de dados, ou estruturas de dados, e dos estados associados a estes dados

i) Coloque nos comentários antes das funções quais são as assertivas do **contrato na especificação**. Diga o que deve ser esperado da função cliente em relação à entrada e o que deve ser garantido pela função servidora na saída.

j) Utilizes iteradores, programação genérica e ponteiros para função.

2) Instrumente o código usando o gcov. Usando o gcov. (<http://gcc.gnu.org/onlinedocs/gcc/Gcov.html>). O makefile deve ser modificado de forma incluir as flags -ftest-coverage -fprofile-arcs. Depois de rodar o executável rode gcov nomearquivo e deverá ser gerado um arquivo .gcov com anotação.

**O gcov é utilizado para saber qual percentual do código é coberto pelos testes. Neste caso os teste devem cobrir pelo menos 80% do código por módulo.**

**Faça a análise estática do programa utilizando o *Splint -weak +infloops +sysdirerrors +compdef*, corrigindo todos os warnings apontados pela ferramenta que analisa os padrões de falta em C.**

## **Módulos**

O programa deve ser dividido em módulos. Os módulos implementados nos laboratórios podem ser utilizados aqui.

## **Engine**

A engine vai ter as funções de interface, inicialização/finalização específicas da **biblioteca ncurses**. A ncurses suporta o uso de interfaces de texto no terminal, para que possamos ter diferentes interfaces como janelas, botões, etc.

Ela pode ser instalada no terminal shell do Linux usando o comando

```
> sudo apt-get install libncurses5-dev
```

Uma vez instalada, a biblioteca ncurses ela pode ser compilada utilizando o GCC e a diretiva `-lncurses`.

Dependendo da instalação ncurses pode ser incluída por `<ncurses/ncurses.h>` ou `<ncurses.h>`.

Existem vários tutoriais disponíveis como :

<http://www.tldp.org/HOWTO/NCURSES-Programming-HOWTO/index.html>

Esse módulo receberá também input do teclado a cada lance. Seguem as funções de Engine:

- void inicia\_ncurses();
- void finaliza\_ncurses();
- int pega\_input(int espera);

### Observações:

Utilize os princípios de modularidade na criação dos módulos, definidos seus módulos de definição, implementação e organizando suas compilações e ligações (**links**) de forma adequada via um único **makefile**.

Na criação dos módulos, lembre-se de usar as diretivas de controle para evitar inclusões múltiplas e identificando também o módulo servidor.

### Controle de qualidade das funcionalidades

Depois de pronto, o aluno deve criar um Makefile. Deve-se criar um *módulo controlador de teste* (disciplinado) usando o CUnit para testar se as principais funcionalidades e restrições dos módulos, atendendo aos critérios da **Interface para linha de Comando**. O teste disciplinado deve seguir os seguintes passos:

- 1) Antes de testar: produzir um roteiro de teste
  - a. definir o contexto (cenário) necessário e selecionar a massa de teste contendo a sequência de ações e valores de teste com os

respectivos resultados esperados e que foi criada segundo um critério de teste.

- 2) Antes de iniciar o teste: estabelecer o cenário do teste
- 3) Criar um módulo controlador de teste, usando a ferramenta **CUnit** para testar as principais funcionalidades de cada módulo.
- 4) Ao testar: produzir um laudo em que todas as discrepâncias encontradas são registradas. Esse laudo pode ser uma saída da execução do CUnit. Somente termine o teste antes de completar o roteiro, caso observe que não vale mais a pena continuar executando o roteiro, uma vez que o contexto para o resto está danificado.
- 5) Após a correção: repetir o teste a partir de **2** até o roteiro passar sem encontrar falhas.

Deve ser gerada uma documentação do código usando o programa DoxyGen (<http://www.stack.nl/~dimitri/doxygen/>): O programa inteiro terá de ser documentado usando DoxyGen. Comentários que vão ficar na documentação devem ser do estilo Javadoc. A documentação deverá ser feita conforme visto em aula e nos laboratórios

Os padrões de codificação e documentação devem estar conforme capítulo 4 e apêndice 4 e 5 do material bibliográfico da disciplina.

## **Parte 2. Escrita**

- Um arquivo **LEIAME.TXT** contendo a explicação de como utilizar o(s) programa(s).
- Tantos arquivos **RELATORIO-nome.TXT** quantos forem os membros do grupo. O tema **nome** deve identificar o membro do grupo ao qual se refere o relatório. Estes arquivos devem conter uma tabela de registro de trabalho organizada como a seguir:

**Data | Horas Trabalhadas | Tipo Tarefa | Descrição da Tarefa Realizada**

Na descrição da tarefa redija uma explicação breve sobre o que o componente do grupo fez. Esta descrição deve estar de acordo com o Tipo Tarefa. Cada Tipo Tarefa identifica uma natureza de atividade que deverá ser discriminada explicitamente, mesmo que, durante uma mesma sessão de trabalho tenham sido realizadas diversas tarefas. Os tipos de tarefa são:

- estudar aulas e laboratórios relacionados
- especificar os módulos

- especificar as funções
- revisar especificações
- projetar
- revisar projetos
- codificar módulo
- Rodar os splint e retirar warnings
- revisar código do módulo
- redigir casos de teste
- revisar casos de teste
- realizar os testes
- instrumentar via gcov
- documentar com Doxygen

Observações:

- Dica: Preencha esta tabela de atividades ao longo do processo. NÃO DEIXE PARA ÚLTIMA HORA, POIS VOCÊ NÃO SE LEMBRARÁ DO QUE FEZ TAL DIA, TAL HORA. Com relatórios similares a esse você aprende a planejar o seu trabalho.

**O trabalho pode ser feito em grupo de no máximo 3.**

**O programa deve ser feito para Linux, utilizado o GCC e GDB na mesma versão do Linf**

**Deve ser enviado um único arquivo compactado (.zip) contendo todos os arquivos necessários (.c .h makefile, estrutura de diretório, informação de como utilizar, etc).**

**Apenas um integrante do grupo deve enviar o trabalho. O nome do trabalho deve ser algo como:**

**TP\_Jose\_12345\_Joao\_54321\_Maria\_12345.zip**

**onde são os primeiros nomes e matriculas dos integrantes do grupo.**

**Cópias de trabalho terão nota zero.**

**Excelente trabalho!**