

Exame Chunin

Christian Luis, José Fortes, Fillype Nascimento,
João Victor, Vinícius Toshiyuki

Prof. Dr. Marcos F. Caetano
CIC116319 Estruturas de Dados, Turma A.
Departamento de Ciência da Computação
Universidade de Brasília

19 de Outubro de 2018

1 Introdução



Chegou a época do ano para os ninja de todo o mundo se reunirem para o **Exame Chunin**. Esse exame é basicamente uma série de testes que *genins* tem que passar para se consagrarem *chunins*. Geralmente a primeira fase é uma prova escrita, a segunda é uma fase de campo e depois existe uma fase

de luta entre os que sobraram para ver quem merece subir de patente no mundo ninja. As fases de prova escrita e de campo já acabaram. Agora você foi convocado para organizar a próxima fase! Com os aprovados da última fase decididos, você terá que gerar lutas e levar o seu preferido à vitória!

2 Objetivos

A representação das lutas será feita utilizando a estrutura de dados do tipo árvore binária. Nesta estrutura, cada nó representará um ninja. A luta acontecerá através da comparação entre as técnicas de dois ninjas, sendo o vencedor aquele que tiver o maior valor para a técnica escolhida. O vencedor do torneio será aquele que conseguir vencer todas as lutas e se mostrar preparado para o título de *chunin*.

2.1 Árvore Binária - Torneio Chunin

A ordem das lutas que fazem parte do torneio é definida pela árvore do tipo binária, formada por 5 níveis. Nesta estrutura, inicialmente, os galhos representam a ordem em que as lutas serão realizadas e os nós do tipo folha representam os ninjas. Neste caso, teremos 16 ninjas diferentes que participarão do exame. Ao final das partidas, o nó raiz representará o grande vencedor.

Cada nó da árvore possuirá uma estrutura do tipo **Ninja *** (vide Seção 2.2). No início do jogo, apenas os nós-folha estarão preenchidos com os ninjas que participarão do exame. As partidas serão realizadas entre 2 nós filhos de um mesmo nó-pai. O vencedor de uma luta passa a ocupar a posição de seu nó-pai e "sobe" na árvore para a próxima etapa, onde enfrentará o outro nó-filho de mesmo pai. Este procedimento será repetido até que o vencedor ocupe a posição do nó-raiz.

Todos os participantes começarão no nível 4 da árvore (*1a Etapa*). Os vencedores da 1ª Etapa serão promovidos para o nível 3 da árvore (*2a Etapa*). Os vencedores das 2ª Etapa irão para o nível 2 da árvore (*3a Etapa*). Os vencedores das 3ª Etapa irão para o nível 1 da árvore (*4a Etapa*). Finalmente, o vencedor da 4ª Etapa irá para o nível 0 da árvore (*Vencedor*) e passa no exame.

2.2 Ninjas/Genins

A árvore binária que representa as lutas do exame chunin é formada pela estrutura do tipo `t_node`, conforme apresentado pela Figura 1. Cada elemento `t_node`, é formado por três ponteiros. Os ponteiros `left` e `right` contém endereços para os elementos da esquerda e direita, respectivamente, caso eles existam. Se o nó representado for do tipo folha, os ponteiros `left` e `right` estarão apontados para `NULL`. O ponteiro `ninja` contém o endereço para estrutura do tipo `Ninja`, conforme definido pela Figura 2. Caso um nó-galho ainda não armazene um `ninja`, o ponteiro `Ninja` apontará para `NULL`.

```
1     typedef struct node {
2         Ninja*  ninja;
3         struct node* left;
4         struct node* right;
5     } t_node;
6
```

Figura 1: Definição da estrutura nó que forma a árvore binária que representa o torneio.

Os ninjas serão representados pela estrutura do tipo `Ninja` (Figura 2). Cada elemento de `Ninja` representa uma técnica do ninja além de um elemento da natureza que o fortalece. Os atributos inteiros (*ninjutsu* (poderes especiais), *genjutsu* (técnicas de ilusão), *taijutsu* (força física) e *defesa*) armazenarão valores entre 0 e 100. Os atributo *nome* e *elemento* são nominais e imutáveis. Os elementos poderão ser Fogo, Vento, Relâmpago, Terra e Água.

```
1     typedef struct {
2         char* nome;
3         char* elemento;
4         int ninjutsu;    // 0 a 100
5         int genjutsu;    // 0 a 100
6         int taijutsu;    // 0 a 100
7         int defesa;      // 0 a 100
8     } Ninja;
9
```

Figura 2: Definição da estrutura `Ninja` que representa um *genin* (ninja).

2.3 Escolhendo os 16 participantes

Para inicialização do jogo, será fornecido um arquivo de entrada (**ninjas.txt**) contendo diversos ninjas que foram prestar o exame. O seu programa deverá ler este arquivo e escolher 16 ninjas, de forma aleatória, para participar das lutas e armazená-los em memória em uma **lista duplamente encadeada**. A escolha aleatória dos ninjas deve ser feita a partir do arquivo de entrada e sem a ajuda de estrutura auxiliar em memória. **Ou seja, os ninjas escolhidos devem ser carregados diretamente para a lista duplamente encadeada**. O uso de estrutura auxiliar (por exemplo, carregar todos os valores do arquivo em memória) fere a ideia básica do uso da lista duplamente encadeada.

Após armazenar os 16 ninjas na lista, você poderá escolher um desses 16 ninjas para jogar por você. Na tela de escolha de personagem, na qual o jogador escolherá um personagem para jogar, o usuário poderá ver apenas **um dos quatro atributos de cada ninja**. A escolha do atributo que estará visível deverá ser feita de maneira aleatória e os ninjas deverão estar listados por ordem em que apareceram na lista duplamente encadeada. É importante destacar que na relação de ninjas que será apresentada ao jogador, o *nome* será omitido. A Figura 6 apresenta um exemplo de como esta relação deverá ser apresentada ao jogador (Seção 3.2).

2.4 Arquivo de Entrada

A árvore binária será preenchida com base no arquivo **ninjas.txt**. Este é um arquivo de texto simples, sem formatação, que apresenta a relação de ninjas que será utilizada na elaboração do torneio. Conforme definição, a árvore binária é formada, inicialmente, por 16 ninjas distintos. Contudo, o arquivo **ninjas.txt** pode conter uma lista muito maior de ninjas possíveis. Cada linha do arquivo representa um ninja distinto, que forma a base de ninjas possíveis a serem utilizados no jogo. No arquivo, cada ninja é descrito da seguinte forma:

```
Naruto, Vento, 42, 33, 55, 74
Sasuke, Fogo, 50, 70, 46, 30
```

Onde, os campos listados por cada linha, segundo a estrutura Ninja (Figura 2), apresentam os seguintes significados:

```
nome, elemento, ninjutsu, genjutsu, taijutsu, defesa
```

2.5 Construindo a Árvore Binária

Conforme descrição apresentada na Seção 2.3, os 16 personagens selecionados de forma aleatória, a partir do arquivo `ninja.txt`, deverão ser armazenados em memória em uma lista duplamente encadeada. Cada elemento da lista armazenará um `ninja`, representado no formato `Ninja` (Figura 2). Esta estrutura será utilizada para a criação da árvore binária.

Conforme descrição apresentada na Seção 2.7, a função `tree_create()` retorna um ponteiro do tipo `t_node` que aponta para o nó raiz da árvore binária, cujos os nós contém o ponteiros `ninja` apontando para `NULL`. Este é o estado inicial do processo de construção da árvore binária. O processo de construção da árvore continua, envolvendo a utilização da lista duplamente encadeada (devidamente inicializada) e a árvore binária retornada pela função `tree_create()`. Neste ponto, o ponteiro `ninja`, da estrutura do tipo `t_node`, de cada **nó folha** da árvore binária, precisa ser apontado para seu respectivo `ninja` (`Ninja`) armazenado na lista duplamente encadeada. Os nós folhas da árvore, da esquerda para direita, devem seguir a mesma ordem dos personagens armazenados na lista duplamente encadeada.

OBS: É importante destacar que todos os algoritmos destinados a manipulação da árvore deverão ser OBRIGATORIAMENTE recursivos.

2.6 Luta

A mecânica da luta do torneio funciona como um *super-trunfo*: você escolhe um atributo (*ninjutsu*, *genjutsu*, *taijutsu* ou *defesa*) para disputar com seu oponente. Vence o lutador que possuir o maior valor do atributo escolhido. Mas antes da comparação dos atributos, o elemento de cada `ninja` da luta deve ser checado. Caso o jogador tenha o elemento diretamente superior ao adversário, o seu atributo vale `x1.2` contra o oponente. Caso o seu adversário tenha vantagem sobre o jogador, os seus atributos são multiplicados por `x0.8`. As outras batalhas (máquina x máquina) não são afetadas pelo elemento. As vantagens elementais acontecem de forma cíclica: Fogo vence Vento, Vento vence Relâmpago, Relâmpago vence Terra, Terra vence Água e Água vence Fogo. As vantagens e desvantagens podem apenas ocorrer com elementos adjacentes ao do jogador, ou seja, se o jogador for do elemento fogo e estivermos batalhando com alguém de terra, os atributos não serão afetados.

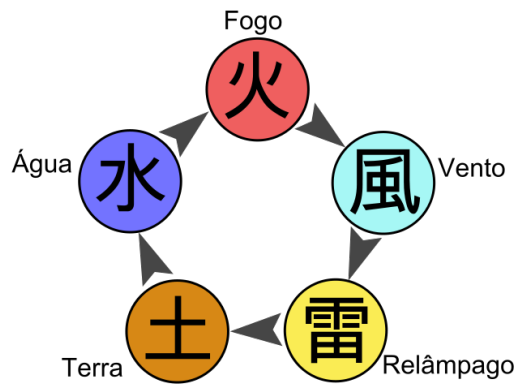


Figura 3: Diagrama de vantagens

Durante a luta, deverá ser mostrado ao jogador todas as informações do seu personagem. Com base nesta informação, o usuário poderá escolher qual atributo irá utilizar para enfrentar o seu oponente.

```

1a ETAPA

Seu personagem: Naruto
1) Ninjutsu : 85
2) Genjutsu : 48
3) Taijutsu : 35
4) Defesa   : 63

Seu adversario: Tobirama

Selecione um atributo: 1
  
```

Figura 4: Tela da Luta

Em toda luta envolvendo o seu personagem, a escolha dos atributos será feita por você. As demais *lutas* do jogo serão controlados pelo "computador". Isto é, o seu programa sorteará de forma aleatória os atributos para que as demais lutas sejam disputadas, definindo assim o vencedor de cada *luta*. Seguindo a definição de ordem apresentada pela árvore binária, a seleção vencedora subirá para o próximo nível do torneio e enfrentará o vencedor da "luta vizinha".

A Figura 4 apresenta um exemplo de como a tela de luta que deverá ser apresentada a cada luta realizada. Conforme pode ser observado, a in-

formação da *etapa* em que a luta está sendo realizada precisa ser apresentada (1a Etapa, 2a Etapa etc.). Em seguida, as informações do seu lutador escolhido (Seção 2.3) serão listadas, bem como, as do seu adversário¹. Note que, do adversário selecionado, somente o nome será apresentado. Por fim, é apresentada a opção de escolha de qual habilidade será utilizada na partida. No exemplo apresentado pela Figura 4, a habilidade escolhida foi *ninjutsu* (opção 1).

Observação: Como regra importante do jogo, NÃO É POSSÍVEL ESCOLHER PARA UMA *LUTA* O MESMO ATRIBUTO UTILIZADO NA *LUTA* ANTERIOR. A ideia aqui é que um mesmo atributo não pode ser utilizado por duas *lutas* consecutivas, uma vez que é necessário uma *luta* de intervalo para o seu reestabelecimento. O valor do atributo usado na *luta* anterior deverá ser representado como inutilizável na *luta* seguinte. Entretanto, o mesmo deve ser disponibilizado novamente após passada uma *luta*.

2.7 Assinatura das Funções a Serem Implementadas

Nesta seção serão relacionadas as assinaturas das funções que deverão ser implementadas neste trabalho.

- Função `t_node* node_create`:

- retorna endereço para estrutura do tipo `t_node` alocada dinamicamente. Os ponteiros `ninja`, `left` e `right` são inicializados com o valor `NULL`;

- `ninja_create()`:

```
Ninja* ninja_create(char* _nome, char* _elemento,
                    int _ninjutsu, int _genjutsu,
                    int _taijutsu, _defesa);
```

- aloca dinamicamente memória para estrutura do tipo `Ninja`. Inicializa **POR CÓPIA** os atributos `nome`, `elemento`, `ninjutsu`, `genjutsu`, `taijutsu` e `defesa` utilizando, respectivamente, os parâmetros `_nome`, `_elemento`, `_ninjutsu`, `_genjutsu`, `_taijutsu` e `_defesa`. Ao final, a função retorna o endereço para estrutura `Ninja` alocada;

¹É importante ressaltar que a ordem das partidas é definida pela árvore binária.

- `void ninja_free(Ninja* ninja);`
 - libera a memória alocada e referenciada por `ninja`;
- `t_node* tree_create();`
 - retorna o endereço para o nó raiz de uma da árvore binária completa de quatro níveis. **TODOS** os nós da árvore apresentam o atributo `ninja` apontado para NULL.
- `void tree_free(t_node* tree);`
 - remove de forma recursiva todos os nós presentes da árvore. **A memória referente ao atributo `ninja` também deve ser liberada;**
- `Ninja* fight(Ninja* ninja_one, Ninja* ninja_two, int attribute);`
 - Compara o valor do atributo definido por `attribute` do `ninja_one` com o do `ninja_two`, retornando o ponteiro para o time vencedor. Em caso de empate, o ponteiro para `ninja_one` deverá ser retornado;
- `void tree_print_preorder(t_node* root);`
 - percorre a árvore binária em pré-ordem, imprimindo os lutadores referenciados em `ninja`;

3 Funcionamento do programa

Nesta seção serão apresentadas algumas telas de funcionamento da mecânica do jogo *Exame Chunin*.

3.1 Tela inicial

Inicialmente, o jogo deverá apresentar uma tela inicial contendo as opções definidas pela Figura 5. Escolhendo a opção ***Iniciar Exame*** o usuário será direcionado para a tela de **Escolha do Personagem** (Seção 3.2). A opção ***Sair*** faz o programa finalizar a sua execução.



Figura 5: Tela inicial do jogo *Exame Chunin*.

3.2 Escolha de personagem

Nesta tela serão apresentados ao jogador os **16 ninjas**, escolhidos de forma aleatória do arquivo **ninjas.txt** e armazenados em memória na estrutura de lista duplamente encadeada. O jogador poderá escolher um dos ninjas relacionados. **É importante destacar que o nome dos ninjas deverá ser omitido e apenas um atributo por ninja deverá ser exibido ao jogador**, conforme apresentado pela Figura 6. A escolha de qual atributo será apresentado é feita de forma aleatória. Após feita a sua escolha, o jogador irá disputar o torneio!

```
Escolha seu personagem

Personagem 1:
Ninjutsu: ?? Genjutsu: ?? Taijutsu: ?? Defesa: 87

Personagem 2:
Ninjutsu: ?? Genjutsu: 16 Taijutsu: ?? Defesa: ??

Personagem 3:
Ninjutsu: ?? Genjutsu: 36 Taijutsu: ?? Defesa: ??

Personagem 4:
Ninjutsu: ?? Genjutsu: ?? Taijutsu: 93 Defesa: ??

Personagem 5:
Ninjutsu: ?? Genjutsu: 22 Taijutsu: ?? Defesa: ??

Personagem 6:
Ninjutsu: ?? Genjutsu: ?? Taijutsu: 28 Defesa: ??

Personagem 7:
Ninjutsu: ?? Genjutsu: ?? Taijutsu: 60 Defesa: ??

Personagem 8:
Ninjutsu: ?? Genjutsu: ?? Taijutsu: ?? Defesa: 27

Personagem 9:
Ninjutsu: 27 Genjutsu: ?? Taijutsu: ?? Defesa: ??
```

Figura 6: Tela para escolha do personagem.

Observação: Por questões de limitação de espaço, a Figura 6 apresenta somente 9 personagens. Contudo, é mandatória a exibição de 16 personagens, conforme especificação.

3.3 Mecânica de Funcionamento do Torneio

A Figura 7 apresenta um exemplo de como é a tela de luta que deverá ser apresentada a cada luta realizada. Conforme pode ser observado, a informação da *etapa* em que a luta está sendo realizada precisa ser apresentada (1a Etapa, 2a Etapa etc). Em seguida, as informações do seu lutador escolhido (Seção 2.3) serão listadas, bem como, as do seu adversário². Note que, do adversário selecionado, somente nome será apresentado. Por fim, é apresentada a opção de escolha de qual habilidade será utilizada no luta. No exemplo apresentado pela Figura 7, a habilidade escolhida foi *ninjutsu* (opção 1).

```
1a ETAPA

Seu personagem: Naruto
1) Ninjutsu : 85
2) Genjutsu : 48
3) Taijutsu : 35
4) Defesa   : 63

Seu adversario: Tobirama
Selecione um atributo: 1
```

Figura 7: Tela de luta da 1ª etapa. O atributo 1 (*ninjutsu*) foi escolhido para o embate.

Finalizada a partida, deverá ser mostrado ao jogador o seu resultado:

```
1a ETAPA: Resultado

VITORIA

Naruto (Ninjutsu 85) x Tobirama (Ninjutsu 47)
Naruto ganhou a batalha

Pressione qualquer teclar para prosseguir
```

Figura 8: Tela de vitória da partida.

²É importante ressaltar que a ordem das partidas é definida pela árvore binária.

Conforme definição (Seção 2.6), na próxima *partida*, o jogador não pode selecionar o mesmo atributo utilizado na partida anterior. A Figura 9 apresenta esta restrição.

```
2a ETAPA

Seu personagem: Naruto
SUPREMACIA ELEMENTAL: Vento > Relampago
Todos os atributos foram multiplicados por x1.2

X) XX      : XX
2) Genjutsu : 57
3) Taijutsu : 42
4) Defesa   : 75

Seu adversario: Kakashi
Selecione um atributo: 4
```

Figura 9: Tela da partida na 2ª etapa. O atributo 1 (*ninjutsu*) não pode ser escolhido pois foi usado na partida anterior. Note que os atributos do jogador foram multiplicados por x1.2 devido ao elemento do jogador ser superior ao do oponente.

```
2a ETAPA: Resultado

VITORIA

Naruto (Defesa 75) x Kakashi (Defesa 47) [x1.2]
Naruto ganhou a batalha

Pressione qualquer teclar para prosseguir
```

Figura 10: Tela de vitória, na 2ª etapa.

De acordo com Figura 11, na 3ª etapa, o atributo que não pode ser escolhido é o 4 (*defesa*), pois foi escolhido na partida anterior. Entretanto, o atributo 1 (*ninjutsu*) que foi utilizado na 1ª etapa já pode ser reutilizado.

```
3a ETAPA

Seu personagem: Naruto
INFERIORIDADE ELEMENTAL: Vento < Fogo
Todos os atributos foram multiplicados por x0.8

1) Ninjutsu : 68
2) Genjutsu : 38
3) Taijutsu : 28
X) XX      : XX

Seu adversario: Itachi

Selecione um atributo: 2
```

Figura 11: Tela da partida na 3ª etapa. O atributo 1 (*ninjutsu*) fica disponível novamente. Note que os atributos do jogador foram multiplicados por x0.8 devido ao seu elemento ser inferior ao do oponente

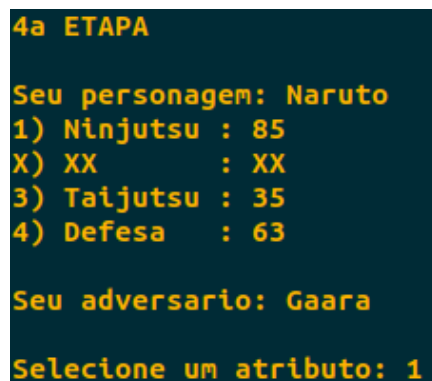
A Figura 12 apresenta a tela quando o valor do atributo escolhido pelo jogador é menor que o valor do seu oponente (derrota).

A partir da tela apresentada pela Figura 12, caso o jogador selecione a opção 1, o programa deve retornar para a tela inicial. Caso seja selecionada a opção 2, o programa deve encerrar sua execução. Além disso, devem ser mostradas todas as lutas que ocorreram **até a etapa em que o usuário perdeu**, no caso do exemplo foram mostradas todas as partidas até a 3ª etapa (etapa na qual o jogador perdeu).



Figura 12: Tela de derrota na 3ª etapa. O jogador tem a opção de voltar ao menu principal (1) ou sair do jogo (2).

Mas é claro que não vamos deixar o Naruto perder nesse exemplo. Digamos que ela tenha vencido o partida anterior e tenha ido para a 4ª etapa. Neste caso, A Figura 13 apresenta o resultado desta continuação.



```
4a ETAPA

Seu personagem: Naruto
1) Ninjutsu : 85
X) XX      : XX
3) Taijutsu : 35
4) Defesa   : 63

Seu adversario: Gaara

Selecione um atributo: 1
```

Figura 13: Tela da partida na final. O atributo 1 (*ninjutsu*) é escolhido para a partida.

Caso o usuário vença a final, **ele será o grande vencedor**. Nessa tela (Figura 14) ele tem a opção de voltar ao menu inicial (1) ou sair do jogo (2). Note que será exibido ao jogador todas as partidas que ocorreram no exame chunin até a vitória. Perceba que existem apenas duas condições em que todas as partidas do torneio aparecem: quando o jogador vence o torneio ou quando ele perde na 4ª etapa (final).

```
4a ETAPA: Resultado

VITORIA

Naruto (Ninjutsu 85) x Gaara (Ninjutsu 73)
Naruto ganhou a batalha

Exame Chunin:
1a ETAPA:
Naruto (Ninjutsu 85) x Tobirama (Ninjutsu 47)
Kakashi (Genjutsu 64) x Sakura (Genjutsu 39)
Itachi (Genjutsu 67) x Madara (Genjutsu 52)
Sasuke (Taijutsu 72) x Deidara (Taijutsu 46)
Gaara (Ninjutsu 73) x Sasori (Ninjutsu 23)
Orochimaru (Defesa 56) x Zetsu (Defesa 48)
Jiraiya (Defesa 62) x Lee (Defesa 36)
Obito (Ninjutsu 43) x Pain (Ninjutsu 32)

2a ETAPA:
Naruto (Genjutsu 75) x Kakashi (Genjutsu 47) [x1.2]
Itachi (Genjutsu 67) x Sasuke (Genjutsu 43)
Gaara (Taijutsu 50) x Orochimaru (Taijutsu 35)
Jiraiya (Taijutsu 56) x Obito (Taijutsu 42)

3a ETAPA:
Naruto (Genjutsu 28) x Itachi (Genjutsu 67) [x0.8]
Gaara (Ninjutsu 73) x Jiraiya (Ninjutsu 48)

4a ETAPA:
Naruto (Ninjutsu 85) x Gaara (Ninjutsu 73)

[1] Voltar ao menu principal
[2] Sair
```

Figura 14: Tela de vitória do Exame Chunin.

4 Avaliação

A avaliação do projeto será composta por duas partes.

4.1 Código (70%)

- (10%) Código modularizado (makefile) e documentado;
- (20%) Implementação da estrutura árvore binária;
- (10%) Implementação da estrutura lista duplamente encadeada;
- (20%) Implementação das partidas;
- (5%) Recuperação das informações contidas no arquivo de entrada (`ninjas.txt`);
- (5%) Implementação de forma correta das funções definidas na Seção 2.7.

4.2 Relatório (30%)

O relatório a ser entregue deve estar OBRIGATORIAMENTE no formato PDF. Ele deve conter as seguintes informações:

- Documentação doxygen
- Descrição de como os problemas foram solucionados e implementados;
- Toda informação necessária para compilação e execução do seu programa.

5 Prazos

Data de entrega: 25/11/2018

Data de apresentações possíveis: 27/11, 29/11, 04/12, 06/12.

Deverão ser marcadas as apresentações no moodle. As vagas serão em ordem de quem pedir primeiro.

6 Observações

As seguintes observações deverão ser consideradas pelos alunos que forem fazer o trabalho:

1. O trabalho deverá ser feito (**OBRIGATORIAMENTE**) individualmente;
2. Para ser considerado entregue, o aluno fará uma apresentação do projeto implementado. No fórum da disciplina serão publicados os horários disponíveis para a apresentação dos trabalhos;
 - **Trabalhos não apresentados ganharão nota zero;**
 - **O aluno** deverá ter o conhecimento de **TODO o trabalho**. Ou seja, ter condições de responder a quaisquer questionamento referente a solução implementada e a documentação feita;
3. Utilize nomes coerentes e auto-explicativos para variáveis, funções e arquivos;
4. Todo arquivo deve conter um cabeçalho explicativo;
5. Seu código deve estar todo comentado;
6. Identação faz parte o código;
 - Será descontado 1,0 ponto do trabalho que apresentar código não indentado;
7. Serão descontados pontos dos trabalhos que apresentarem vazamento de memória (utilize o *valgrind* para verificação);
8. O trabalho deve rodar **OBRIGATORIAMENTE** na plataforma GNU/Linux;
 - Trabalhos que não rodarem na plataforma GNU/Linux levarão nota zero;
9. Deve ser utilizado **OBRIGATORIAMENTE** a sintaxe ANSI C;
10. No relatório do trabalho deve conter as instruções para compilação do código e a relação de todas as bibliotecas utilizadas;

- A nota do trabalho que não compila é **ZERO**. É sua responsabilidade se certificar que o código submetido está correto e compila na plataforma GNU/Linux;
11. O relatório deverá ser entregue **OBRIGATORIAMENTE** no formato de arquivo PDF;
 12. Códigos copiados (entre alunos, retirados da Internet ou do repositório da disciplina) serão considerados "cola" e todos os alunos envolvidos ganharão nota zero;
 13. Dúvidas sobre o trabalho deverão ser tiradas **NO FÓRUM DE DÚVIDAS DO AMBIENTE APRENDER**. Desta forma, dúvidas comuns e esclarecimentos poderão ser respondidos uma única vez para toda a turma.