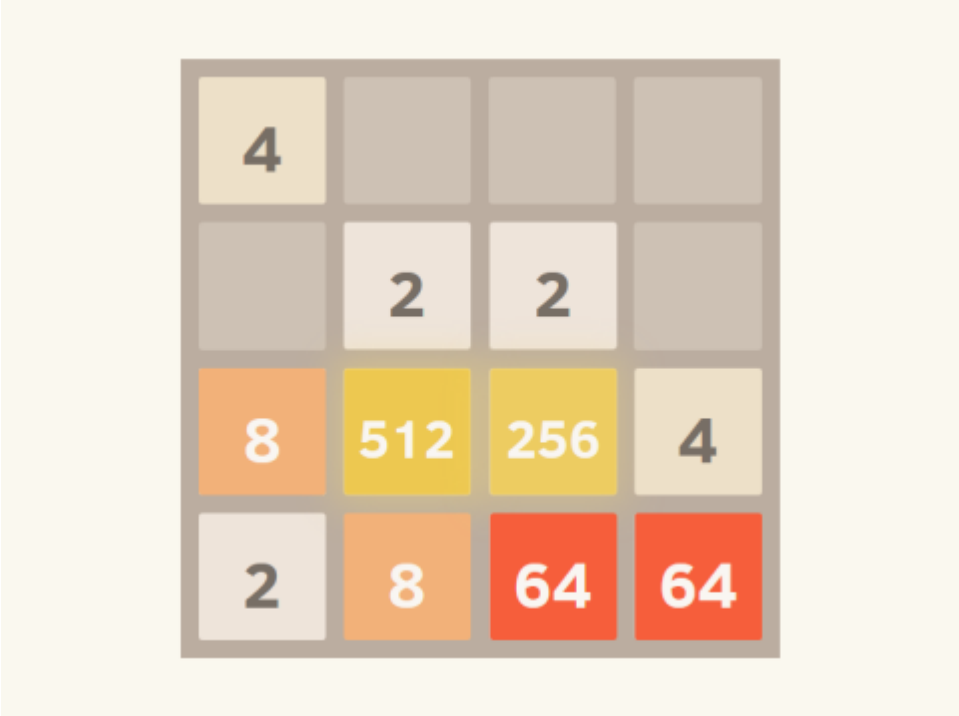
2048

DESCRIÇÃO DE TRABALHO PRÁTICO



Introdução

O início de 2014 foi marcado pelo sucesso de um jogo Indie simples, mas que atraiu um grande número de adeptos: 2048. É um jogo de raciocínio baseado em turnos que ocorre em um tabuleiro 4x4, onde valores (2 ou 4) são inseridos de forma aleatória a cada turno. O jogador precisa, então, controlar a movimentação das peças sobre o tabuleiro deslocando-os em uma das quatro direções (cima, baixo, esquerda ou direita). Sempre que dois valores iguais se encontram em função da movimentação realizada pelo jogador, seus valores são somados. Isso faz as duas casas ocupadas pelos valores no tabuleiro passem a ocupar apenas uma, contendo o valor somado. Como os valores iniciais são sempre 2 ou 4, os valores seguintes são obrigatoriamente uma potência de 2 (4, 8, 16, 32, 64,...). O objetivo do jogador é, dentro do espaço limitado do tabuleiro e atrapalhado pela aleatoriedade dos valores inseridos a cada turno, conseguir alcançar, através de subsequentes somas, o valor 2048.

Para compreender melhor a dinâmica aqui descrita, pode-se testar o jogo online (para quem ainda não o conhece) no endereço: http://gabrielecirulli.github.io/2048/

Descrição da atividade

A atividade deve ser realizada em **duplas**, cuja tarefa é construir em C um jogo semelhante ao 2048, porém em modo texto (a ser executado em linha de comando). Uma possível *interface* é apresentada a seguir. Porém, cada dupla pode definir sua própria interface, desde que atenda os critérios descritos na atividade.

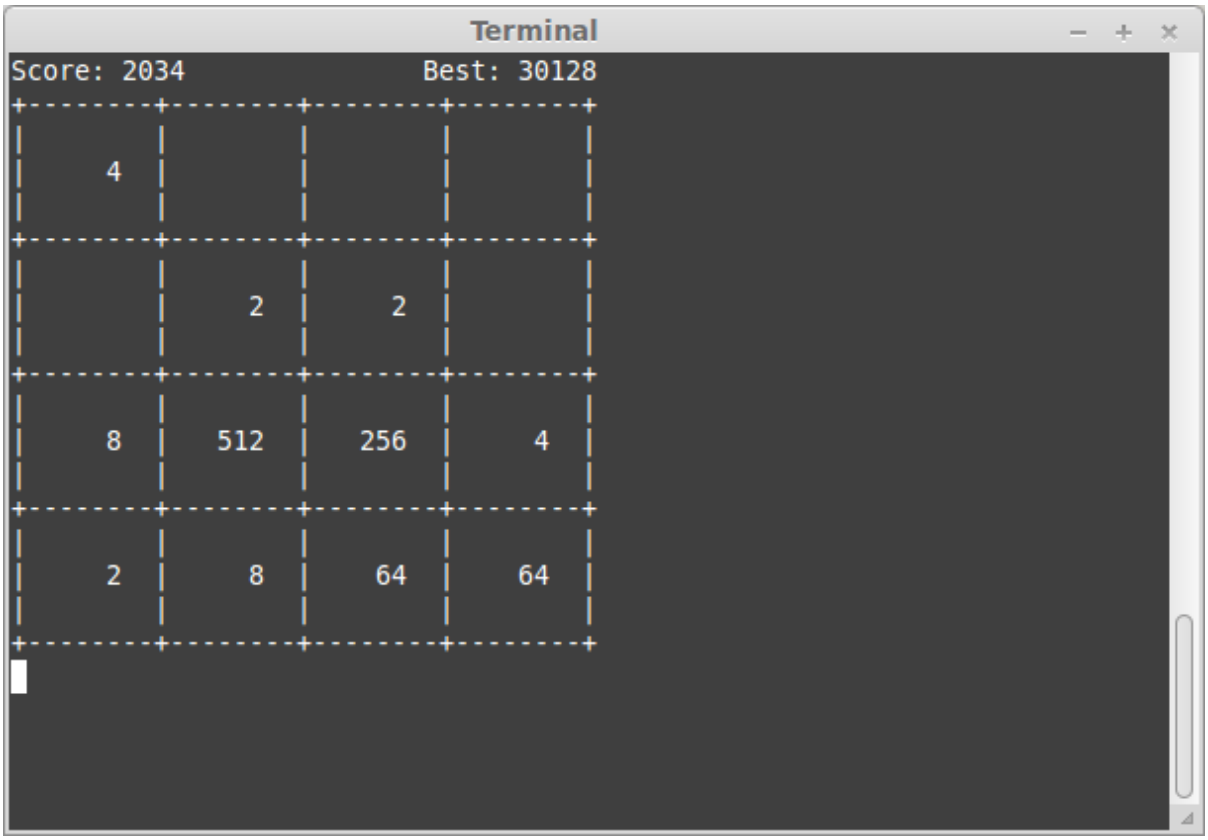


Figura 1 - Exemplo de interface do 2048 no terminal (linha de comando).

Além das características básicas da mecânica do jogo, ele deve atender os seguintes requisitos funcionais:

- A pontuação atual (Score) e a mais alta já alcançada (Best) devem ser apresentadas junto com o tabuleiro durante o jogo;
- O programa deve gravar em arquivo a pontuação mais alta para poder apresentá-la em execuções subsequentes;
- O jogador poderá salvar o estado do tabuleiro para poder continuar o jogo em uma execução futura.

A leitura do teclado sem apresentar *echo* (sem aparecer o caractere digitado na tela) e sem precisar digitar *enter* não é um requisito da atividade. Ou seja, você pode usar mecanismos simples como scanf() ou getchar(). O uso de bibliotecas avançadas para esse fim pode valer ponto extra, a depender da complexidade envolvida (ver seção de pontuação).

Além desses requisitos, o programa deve atender os seguintes critérios de programação:

- Uso de matrizes (arranjos de arranjos);
- Uso de registros (struct);
- Definição de novos tipos de dados através de typedef;
- Modularização do programa em diferentes arquivos (uso de arquivos .c e .h);
- Identação adequada do código-fonte seguindo um padrão (seja ele qual for);
- Documentação adequada do código-fonte.

Entrega do trabalho

Além do código-fonte (arquivos .c e .h), a dupla deve igualmente fornecer um arquivo README.txt contendo:

- O nome dos integrantes da dupla;
- A descrição do que foi feito (incluindo funcionalidades extras), do que faltou fazer (em caso de trabalho incompleto) e do que fariam diferente se refizessem o trabalho novamente (em C);
- A descrição do trabalho desenvolvido por cada integrante (se fizeram juntos e quais partes do código cada um contribuiu), bem como que partes foram copiadas/adaptadas de códigos na internet (indicar a fonte).

Pontuação da atividade

O trabalho aqui descrito valerá como nota da terceira avaliação de PITP e até dois pontos extras a serem adicionados em uma das avaliações de ITP. Para isso, cada integrante da dupla deve indicar em qual avaliação de ITP (1a, 2a ou 3a) os pontos extras serão creditados.

A pontuação da avaliação seguirá os critérios e distribuição abaixo:

- **Atendimento dos requisitos funcionais: 60%**
  a mecânica do jogo está correta? está completa? o jogo salva pontuação máxima? o jogo salva estado do tabuleiro? etc.
- **Uso dos recursos da linguagem C: 30%**
  a dupla demonstrou saber usar de forma adequada os recursos da linguagem C (arranjos, structs, typedefs, laços, etc)?
- **Organização do código e documentação: 10%**
  o código está documentado? a identação e uso de { } seguem um padrão? o programa está devidamente modularizado em diferentes arquivos?

Pontuação extra será dada para trabalhos que forem além dos requisitos descritos neste documento, isso inclui, mas não está limitado a:

- Uso de bibliotecas de *interface*, seja em modo texto (ncurses) ou gráficas (Gtk, Qt, Allegro, etc);
- Extrapolação da mecânica do jogo, criando um novo jogo similar, mas com complexidade maior que o 2048.

A pontuação a ser dada pelas funcionalidades extras não é definida *a priori*. Cada caso será avaliado em função da complexidade envolvida. Itens extras de baixa complexidade serão desconsiderados na avaliação.

A atividade será avaliada ao longo das aulas práticas de PITP. Isso significa que as duplas deverão apresentar ao professor a evolução do código a cada aula prática. Soluções que "miraculosamente" aparecem "do nada" no dia da apresentação do trabalho não serão consideradas. É importante, então, que a dupla venha às aulas práticas pois o professor irá acompanhar o desenvolvimento da dupla.

Observações adicionais:

1. É fácil encontrar soluções para esta atividade na internet. As soluções podem ser estudadas, mas não copiadas. Caso a dupla use trechos de programas encontrados na internet (por exemplo, para leitura de uma tecla digitada sem apresentá-la no terminal), a dupla deve indicar isto no arquivo README.txt. Caso haja cópia sem haver menção, o trabalho será anulado. (Assim como é fácil para os alunos encontrarem soluções para o problema, também o é para os professores).

2. Apesar do trabalho ser em dupla, a **nota é individual**. Isso significa que os alunos serão avaliados individualmente ao longo das aulas de laboratório, podendo um integrante ficar com 10,0 enquanto o outro com 0,0 (se o professor identificar que este não fez nada).