

Universidade de Brasília
Departamento de Ciência da Computação
Disciplina: Métodos de Programação
Código da Disciplina: 201600

Métodos de Programação - 201600

Trabalho 2

O objetivo deste trabalho é utilizar o desenvolvimento orientado a testes (TDD) para resolver o problema de verificar o jogo da velha. As regras do jogo da velha são dadas em: <http://portaldoprofessor.mec.gov.br/fichaTecnicaAula.html?aula=28141>

O jogo da velha é representado como uma matriz 3x3 de inteiros.

O valor 0 significa que a posição está vazia

O valor 1 significa que a posição está com um X

O valor 2 significa que a posição está com um O

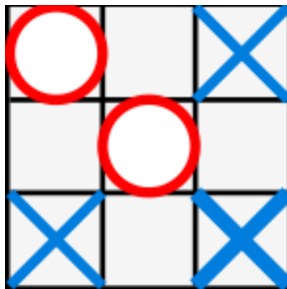
Ex. A matriz

[2, 0, 1]

[0, 2, 0]

[1, 0, 1]

Representa o jogo:



O objetivo é fazer e testar uma função que:

Tem como parâmetro a matriz 3x3 inteiros

Retorna 1 se o vencedor foi o X

Retorna 2 se o vencedor foi o O

Retorna 0 se o jogo está empatado

Retorna -1 se o jogo está indefinido (ex. tem apenas um X)

Retorna -2 se o jogo é com certeza impossível pelas regras (ex. todas as posições são X)

O desenvolvimento deverá ser feito passo a passo seguindo a metodologia TDD. A cada passo deve-se pensar qual é o objetivo do teste e o significado de passar ou não no teste.

1) O programa deverá ser dividido em módulos e desenvolvido em C ou C++. Deverá haver um arquivo `velha.c` (ou `.cpp`) e um arquivo `velha.h` (ou `.hpp`). Deverá haver também um arquivo `testa_velha.c` (ou `.cpp`) cujo objetivo é testar o funcionamento da biblioteca de verificação do jogo da velha.

Faça um Makefile utilizando o exemplo de makefile 5 dado em:

(<http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/>)

Não se esqueça de criar os diretórios correspondentes.

O programa e o módulo devem ser depurados utilizando o GDB.

(<http://heather.cs.ucdavis.edu/~matloff/UnixAndC/CLanguage/Debug.html>)

(<https://www.cs.umd.edu/~srhuang/teaching/cmssc212/gdb-tutorial-handout.pdf>)

2) Utilize o padrão de codificação dado em: <https://google.github.io/styleguide/cppguide.html> quando ele não entrar em conflito com esta especificação. O código deve ser claro e bem comentado. O código deve ser verificado se está de acordo com o estilo usando o `cpplint` (<https://github.com/cpplint/cpplint>).

Utilize o `cpplint` desde o início da codificação pois é mais fácil adaptar o código no início.

3) Faça um documento dizendo quais testes você fez a cada passo e o que passar neste teste significa.

4) O desenvolvimento deverá ser feito utilizando um destes frameworks de teste:

4.1) `gtest` (<https://code.google.com/p/googletest/>)

4.2) `catch` (<https://github.com/philsquared/Catch/blob/master/docs/tutorial.md>)

5) Deverá ser entregue o histórico do desenvolvimento orientado a testes feito através do `github` (<https://github.com/>)

6) Faça a análise estática do programa utilizando o `cppcheck`, corrigindo os erros apontados pela ferramenta.

Utilize `cppcheck --enable=warning` .

para verificar os avisos nos arquivos no diretório corrente (.)

Utilize o cppcheck sempre e desde o início da codificação pois é mais fácil eliminar os problemas logo quando eles aparecem. Devem ser corrigidos apenas problemas no código feito e não em bibliotecas utilizadas (ex. gtest, catch)

7) É interessante utilizar o Valgrind (valgrind.org/).

Devem ser enviados para a tarefa no ead.unb.br um arquivo zip onde estão compactados todos os diretórios e arquivos necessários. O documento deve estar na raiz do diretório. Todos os arquivos devem ser enviados compactados em um único arquivo (.zip) e deve ser no formato matricula_primeiro_nome ex: 06_12345_Jose.zip. Deve conter também um arquivo leiametext que diga como o programa deve ser compilado.

Data de entrega:

28/ 9 /19

Pela tarefa na página da disciplina no ead.unb.br