



Trabalho Prático 1 (TP1) - 10 pontos, peso 1.

- Data de entrega: 17/04/2022 até 23:55. O que vale é o horário do `run.codes`, e não do *seu*, ou do *meu* relógio!!!
- O padrão de entrada e saída deve ser respeitado exatamente como determinado no enunciado.
- Parte da correção é automática, não respeitar as instruções enunciadas pode acarretar em perda de pontos.
- O trabalho é em grupo de até 3 (três) pessoas.
- Entregar um relatório.
- Procedimento para a entrega:
 1. Submissão: via **run.codes**.
 2. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
 3. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
 4. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos `.h` e `.c` sempre que cabível.
 5. Os arquivos a serem entregues, incluindo aquele que contém `main()`, devem ser compactados (`.zip`), sendo o arquivo resultante submetido via **run.codes**.
 6. Dentre os arquivos submetidos, deve existir um intitulado `compilcao.txt`, contendo os comandos especificados no `prompt/console` para compilar e executar seu programa.
 7. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.
- *Bom trabalho!*

1 Objetivos

Este trabalho prático tem como objetivo principal fundamentar conceitos aprendidos em sala e nas práticas. Entre os objetivos, pode-se destacar:

- Implementar um tipo abstrato de dados.
- Calcular complexidade de operações simples.
- Virar um ninja na verificação de um Sudoku.

	1	2	3	4	5	6	7	8	9
1		1	4	5	2				9
2	3				9			4	5
3	5		8					6	7
4						3	7		
5			6	2		5	1		
6			3	4					
7	4	3					6		2
8	8	2	7		4				5
9	1				5	2	8	4	

Figura 1: Exemplo tabuleiro Sudoku

2 Descrição do Problema

Neste trabalho prático você deverá implementar um verificador de soluções para o jogo Sudoku e também um aconselhador de valores válidos¹. No Sudoku, um tabuleiro vem parcialmente preenchido e você deve encontrar números que o completem, mantendo linhas, colunas e regiões válidas: sem números repetidos. Uma célula específica do jogo será referenciada pela sua linha e coluna. Por exemplo, a célula destacada em azul é a (3, 6). A região em verde é a 6, e para que a solução seja válida as células vazias não podem ser preenchidas com 1 e 7. Para que a linha 2 (em rosa) seja válida, as células vazias só poderão ser preenchidas com 2, 6, 7 ou 8. Já na coluna 3 (amarela), só podemos colocar 1, 2, 5 ou 9.

3 Imposições e comentários gerais

Neste trabalho, as seguintes regras devem ser seguidas:

- Seu programa não pode ter *memory leaks*, ou seja, toda memória alocada pelo seu código deve ser corretamente liberada antes do final da execução. (Dica: utilize a ferramenta *valgrind* para se certificar de que seu código libera toda a memória alocada)
- Um grande número de *Warnings* ocasionará a redução na nota final.

3.1 Comentários Gerais:

- Clareza, identificação e comentários no código também vão valer pontos. Por isso, escolha cuidadosamente o nome das variáveis e torne o código o mais legível possível.
- Trabalhos copiados (e FONTE copiado) terão nota zero, além de os alunos envolvidos no plágio perderem toda a nota atribuída a participação e pontos extras.
- O trabalho é individual e o professor pode realizar uma entrevista com o aluno sobre o mesmo.

3.2 O que deve ser entregue

- Código fonte do programa em C (bem indentado e comentado).
- Documentação do trabalho (relatório). A documentação deve conter:
 1. **Implementação:** descrição sobre a implementação do programa. Não faça “print screens” de telas. Ao contrário, procure resumir ao máximo a documentação, fazendo referência ao que julgar mais relevante. É importante, no entanto, que seja descrito o funcionamento das principais

¹Se você nunca jogou Sudoku (o que eu duvido), então leia sobre o assunto na rede. Há vários sites e apps explicando as regras e inclusive permitindo que você jogue online (por exemplo, <http://www.sudoku.name/index-pt.php>).

funções e procedimentos utilizados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Muito importante: os códigos utilizados na implementação devem ser inseridos na documentação.

2. **Impressões gerais:** descreva o seu processo de implementação deste trabalho. Aponte coisas que gostou bem como aquelas que o desagradou. Avalie o que o motivou, conhecimentos que adquiriu, entre outros.
3. **Análise:** deve ser feita uma análise dos resultados obtidos com este trabalho.
4. **Conclusão:** comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
5. **Formato:** PDF ou HTML.

3.3 Como deve ser feita a entrega

Verifique se seu programa compila e executa na linha de comando antes de efetuar a entrega. Quando o resultado for correto, entregue via *run.codes* até a 17/04/2022 até 23:55 um arquivo **.ZIP** com o nome e sobrenome do aluno. Esse arquivo deve conter: (i) os arquivos *.c* e *.h* utilizados na implementação, (ii) instruções de como compilar e executar o programa no terminal, e (iii) o relatório em **PDF**.

Exemplo de um nome do arquivo a ser entregue: *pedro-silva.zip*.

4 Detalhes da implementação

Para atingir seu objetivo, você deverá construir um Tipo Abstrato de Dados Tabuleiro como representação do Sudoku que você quer resolver. O TAD deverá implementar, pelo menos, as seguintes operações:

1. **void** TabuleiroInicializa(NomeArquivo): inicializa tabuleiro a partir de arquivo.
2. **Celula[]** defineVazias(Tabuleiro): retorna célula vazias do tabuleiro.
3. **boolean** EhValido(Tabuleiro): verifica se uma célula é válida.
4. **int[]** valoresValidos(Tabuleiro, Celula): retorna todos os valores válidos para uma célula vazia.

onde *Celula* é o par (linha, coluna). O TAD deve ser implementado utilizando a separação interface no *.h* e implementação *.c* discutida em sala, bem como as convenções de tradução. Caso a operação possa dar errado, devem ser definidos retornos com erro, tratados no corpo principal.

Você deve definir e implementar o *main.c* (corpo principal do seu programa) e outras operações para seu TAD. A implementação da Estrutura de Dados do TAD Tabuleiro deverá necessariamente utilizar um arranjo de duas dimensões para representar ou o tabuleiro completo ou as regiões, você escolhe.

Abaixo está um exemplo de como pegar o nome do arquivo pelo terminal.

```
1 int main(int argc, char** argv){
2
3     if (argc < 2){
4         printf("Uso: %s <arquivo_tabuleiro>\n", argv[0]);
5         return 0;
6     }
7     // o nome do arquivo esta em argv[1]
8
9     return 0;
10 }
```

4.1 Entrada

Os tabuleiros serão informados para seu programa por meio de arquivos. O nome do arquivo deverá ser informado na linha de comando e seguir o formato (entradas vazias são representadas por 0) onde há uma linha do arquivo por linha do tabuleiro. O tabuleiro acima seria representado como:

sud1.txt

```

0 1 4 5 2 0 0 0 9
3 0 0 0 9 0 4 5 1
5 0 8 0 0 0 0 6 7
0 0 0 0 0 3 7 0 0
0 0 6 2 0 5 1 0 0
0 0 3 4 0 0 0 0 0
4 3 0 0 0 0 6 0 2
8 2 7 0 4 0 0 0 5
1 0 0 0 5 2 8 4 0

```

4.2 Saída

Seu objetivo é construir um programa para verificar se um preenchimento (parcial ou completo) de um tabuleiro é válido e, caso afirmativo, para cada célula vazia, indicar quais são os valores válidos para mesma. Se o tabuleiro é inválido, todas as inconsistências deverão ser apontadas. O tabuleiro será lido de um arquivo de entrada, fornecido como parâmetro:

Terminal

```
.\sudoku <arquivo_entrada>
```

A saída dependerá do estado do tabuleiro e deverá seguir exatamente os formatos sugeridos abaixo.

1. O tabuleiro completo (todas as células preenchidas) e válido (sem valor duplicado na linha, coluna ou região), só informe.

6	1	4	5	2	7	3	8	9
3	7	2	6	9	8	4	5	1
5	9	8	1	3	4	2	6	7
2	5	1	8	6	3	7	9	4
9	4	6	2	7	5	1	3	8
7	8	3	4	1	9	5	2	6
4	3	5	9	8	1	6	7	2
8	2	7	3	4	6	9	1	5
1	6	9	7	5	2	8	4	3

Terminal

```
Jogo completo. Voce ganhou!
```

2. O tabuleiro incompleto e válido: informe os valores possíveis para as células vazias.

6	1	4	5	2	7	3	8	9
3	7	2	6	9	8	4	5	1
5	9	8		3		2	6	7
2	5	1	8	6	3	7	9	4
9	4	6	2	7	5	1	3	8
7	8	3			9	5	2	6
4	3	5	9	8		6	7	2
8	2	7	3		6	9	1	5
1	6	9	7	5	2	8	4	3

Terminal

Voce esta no caminho certo. Sugestoes:

(3,4): 1 4

(3,6): 1 4

(6,4): 1 4

(6,5): 1 4

(7,6): 1

(8,5): 4

3. O tabuleiro inválido (completo ou não): você deverá indicar TODAS as inconsistências.

6	1	4	5	2	7	3	8	9
3	7	2	6	9	8	4	3	1
5	9	8	1	3	4	2	6	7
5	5	1	8	6	3	7	9	4
9	4	6	2	7	5	1	3	8
7	8	3	4	1	9	5	2	6
4	3	5	9	8	1	6	7	2
8	2	7	3	4	6	9	1	5
1	6	9	7	5	2	8	4	3

Terminal

Alguma coisa deu errado... Invalidos:

Linha 2: (2,1) e (2,8)

Linha 4: (4,1) e (4,2)

Coluna 1: (3,1) e (4,1)

Coluna 8: (2,8) e (5,8)

Regiao 3: (1,7) e (2,8)

Regiao 4: (4,1) e (4,2)

A SAÍDA DA SUA IMPLEMENTAÇÃO DEVE SEGUIR EXATAMENTE A SAÍDA PROPOSTA.

5 PONTOS EXTRAS

Será concedido 0,1 extra para quem gerar o relatório em Latex (deve ser enviado os *.tex*).

Será concedido 0,4 extra para quem criar uma abordagem para solucionar um tabuleiro de sudoku. Você deve adicionar ao relatório, a descrição da abordagem proposta e enviar os fontes.