RELATÓRIO SUDOKU

João Vitor Alves Gonçalves

Pedro Henrique Kauan Alonso Mendes

12.11.2024

ALGORITMOS E PROGRAMAÇÃO II

1. INTRODUÇÃO

Inicialmente, foi feita uma pesquisa sobre o funcionamento do jogo Sudoku, para entender suas regras e características. Além disso, consultou-se o arquivo PDF disponibilizado no AVA, que serviu como referência para definir as funcionalidades obrigatórias e opcionais do código. A partir dessas informações, discutiu-se como o código deveria ser estruturado e organizado.

1.1. Organização

Primeiramente, foi criado um repositório no GitHub para versionamento do código, visando evitar perda de informações durante o desenvolvimento do trabalho. Em seguida, as funcionalidades do projeto foram divididas, segmentando os problemas para facilitar a implementação de cada parte individualmente. Após concluídas, as funções foram integradas em um único código. Dessa forma, foram criados dois arquivos .cpp:

- **carregue.cpp**: contém a função carregue, responsável por toda a lógica da opção 1 do menu principal.
- **validacaoDePosicao.cpp**: implementa toda a lógica para a validação de posições nas linhas, colunas e quadrantes.

1.2. Ambiente de desenvolvimento

Durante o desenvolvimento do trabalho, foram utilizados quatro computadores, cada um com um compilador diferente. No entanto, todos usaram o mesmo editor de texto: Visual Studio Code.

- Computador 1: Linux Ubuntu, Compilador clang versão 18.1.3
- Computador 2: Linux Manjaro, Compilador g++, versão 14.2.1
- Computador 3:Windows 11, Compilador gcc, versão 11.5
- Computador 4:Windows 11, Compilador gcc, versão 14.2

2. IMPLEMENTAÇÕES APLICADAS

Como mencionado anteriormente, a organização do código foi feita segmentando os problemas em partes menores, o que torna o código mais fácil de modificar.

2.1. Carregue.cpp

A função carregue exige duas strings para os nomes do arquivo de texto e do arquivo binário. Para facilitar a manipulação dessas strings, optou-se por utilizar a biblioteca <string.h>, que permite lidar mais facilmente com a entrada obtida por fgets. Dessa forma, o usuário só precisa digitar o nome do arquivo, sem incluir a extensão. Essa funcionalidade também é usada na função crie_arquivo_binario, onde são gerados três caracteres aleatórios para o nome, e a extensão do arquivo binário é adicionada automaticamente com a ajuda da biblioteca.

2.2. validacaoDePosicao.cpp

Na validação de posição, foi implementada uma estratégia de verificação com um loop for, que verifica se o valor já existe na linha e na coluna. Para a validação do quadrante, foram usadas as funções para determinar o início e o fim de x e y, possibilitando que dois loops for percorressem o quadrante 3x3 e verificassem a existência do valor.

2.3. Resolver um passo

Foi implementado a estratégia de verificar se existe uma posição vazia, se existir essa posição é descoberta através de dois for percorrendo o quadro inteiro até achar a posição com valor igual a 0, com a coordenada da célula vazia é testado valores de 1 a 9 até encontrar um valor que não infringe alguma regra do sudoku

2.4. Resolver o jogo completo

Nesta função primeiramente é encontrado alguma célula vazia, se não for encontrada o sudoku está completo, se for encontrada então usamos a mesma estratégia da funcao resolver um

passo, entretanto chamamos recursivamente a função resolver o jogo completo e assim sucessivamente, se posteriormente em algum momento o jogo ficar travando entao na célula inicial é colocado outro valor e assim chamando recursivamente novamente a função até o jogo estar completo.

3. PROBLEMAS ENFRENTADOS

3.1. JUNÇÃO DOS CÓDIGOS

Houveram diversos problemas para a execução do trabalho, um deles foi a junção de todas as funções em um só código tendo em vista que no código principal não tinha em diversos momentos alguma variável declarada ou algum erro de carácter presente no código separado e que não havia no código principal.

3.2. COMPILADORES DIFERENTES

O maior problema enfrentado foi a diferença de compilador e computadores, várias vezes o código não compila em uma máquina e em outra sim. Um exemplo a ser dito é o modo de abertura de um arquivo no computador 1, tanto o compilador clang e o g++ nao permitiam o código funcionar devido o modo de abertura do arquivo binário ser rb+ o código funcionou apenas ao mudar para wb+, entretanto nos demais computadores o rb+ funcionava normalmente.

3.3. COMO RESOLVER O JOGO AUTOMATICAMENTE

Foi gasto muito tempo para a implementação das funções de resolver um passo e resolver o jogo completo. Principalmente resolver jogo completo pois testar até achar um valor válido não é a melhor estratégia tendo em vista que é apenas uma solução momentânea sendo possível mais a frente o jogo ficar travado por ter inserido qualquer valor e não um valor pensado. Sendo assim foi pesquisado algoritmos de resolver sudoku e foi encontrado o *backtracking* que basicamente é um algoritmo que resolva o sudoku a partir da tentativa e erro podendo ser chamado “no braço”. logicamente o algoritmo testa um número em uma célula e, caso viole as restrições (como duplicação em linhas, colunas ou subgrades), volta à célula anterior para tentar outro número. Este processo continua até encontrar uma solução completa e válida para o quebra-cabeça.O

algoritmo foi estudado e escrito através da linguagem de programação C e aplicado para sudoku. O principal artigo usado se encontra em https://pt.wikipedia.org/wiki/Algoritmo_de_sudoku.