Relatorio do Trabalho de Implementação 1

Grupo: Caio Silas de Araujo Amaro(21.1.4111),Henrique Dantas Pighini(21.1.4025)

17 de Abril,2022

1 Implementação

Começando pela implementação, fizemos um código simples porém que dá conta do serviço. Construímos 3 structs para ficar mais fácil na resolução dos problemas a nós apresentados:

```
struct sudoku{
TCelula** celulas;
TRegiao** regiao;
int vazias;
int situacao;
};

struct regiao{
int regiao;
TCelula** celulasRegiao;
};

struct celula{
int linha;
int coluna;
int valor;
int* possiveis;
int regiao;
};
```

Assim sendo, a partir de uma célula eu conseguiria saber a região que ela se encontrava e em que posição do sudoku a mesma estava facilitando muito a minha manipulação desses valores no código.
Após construir essas Structs construímos funções simples para alocar elas e fomos para as funções que eram requisitos para nosso código segundo o pdf do TP.

Para realizar o que nos foi pedido utilizamos apenas as funções do pdf, começando pela void TabuleiroInicializa(char *arquivo, TSudoku *sudoku){
que em resumo inicializa nosso sudoku a partir de um arquivo texto recebido pelo main através de argv.

```
FILE* arq = fopen(arquivo, "r");
if(arq == NULL){
printf("Nao foi possivel abrir o arquivo!!");
exit(0);
}

//Preenchendo o sudoku a partir do arquivo
for(int i = 0; i ¡ 9; i++){
for(int j = 0; j ¡ 9; j++){
fscanf(arq, "%d", &sudoku-¿celulas[i][j].valor);
sudoku-¿celulas[i][j].linha = i;
sudoku-¿celulas[i][j].coluna = j;
```

}
}

Depois de você inicializar o sudoku o próximo passo é ver quantas células vazias o mesmo tem, assim criamos a 2° função:
TCelula* defineVazias(TSudoku *tabuleiro){

Está sendo bem simples, já que sua única função é passar pelo tabuleiro e pegando as células com valor 0 e as inserindo em um vetor de células.

for(int i = 0; i ¡ 9; i++){
for(int j = 0; j ¡ 9; j++){
if(tabuleiro-¿celulas[i][j].valor == 0){
aux[n] = tabuleiro-¿celulas[i][j];
n++;
}
}
}

return aux;

Após isso nós criamos a 3° função que verifica a situação do sudoku para ver se ele é válido:
int EhValido(TSudoku *tabuleiro){

Essa função verifica a partir de uma célula se esta é válida na linha, coluna e região que esta mesmo pertence. (Função é demasiada grande então mostrarei apenas uma das sequências de for)
int flag = 0;
//Verifica inconsistências nas linhas
for(int i = 0; i ¡ 9; i++){
for(int j = 0; j ¡ 8; j++){
for(int k = j+1; k ¡ 9; k++){
if(tabuleiro-¿celulas[i][j].valor == tabuleiro-¿celulas[i][k].valor tabuleiro-¿celulas[i][j].valor != 0){
if(flag == 0){
printf(”Alguma coisa deu errado... Invalidos:”);
flag++;
}

printf(”Linha %d: (%d,%d) e (%d,%d)”, i+1, i+1, j+1, i+1, k+1);
}

}
}
}

A última função requisito é :
int* valoresValidos(TSudoku *tabuleiro, TCelula celula){

Que a partir do vetor de posições vazias gerado na 2° função retorna os valores possíveis que a célula pode assumir :

//Verifica possibilidades nas linhas
for(int j = 0; j ¡ 9; j++){

```
if(tabuleiro-¿celulas[celula.linha][j].valor != 0){ valores[tabuleiro-¿celulas[celula.linha][j].valor-1]++;
}
}
```

E após descobrir todos os valores possíveis eu criei uma função para averiguar os dados recebidos pelas funções e retornar a situação do sudoku

```
int situacaoSudoku(TSudoku *tabuleiro, TCelula *vazias, int valido){
int *validos;

if(vazias[0].valor == -1  valido == 0){
printf("Jogo completo. Voce ganhou!");
return 0;
}
else if(valido == 0){
printf("Voce esta no caminho certo. Sugestoes:");
for(int i = 0; i ¡ tabuleiro-¿vazias; i++){
validos = valoresValidos(tabuleiro, vazias[i]);
if(validos == NULL){
exit(0);
}
printf("(%d,%d):", vazias[i].linha + 1, vazias[i].coluna + 1);

for(int j = 0; j ¡9; j++){
if(validos[j] == 0){
printf("%d ", j+1);
}
}
printf("");
validos = NULL;
}
}
free(validos);

    return 1;
}
```

E assim eu finalizo o meu código chamando todas essas funções no main e lidando com os dados retornados por elas.

```
include ¡stdio.h¿
include ¡stdlib.h¿
include ¡string.h¿
include "funcoes.h"


int main (int argc, char** argv){


TSudoku *sudoku;
AlocarSudoku(sudoku);
```

```
TabuleiroInicializa(argv[1], sudoku);

TCelula *vazias = defineVazias(sudoku);
int valido = EhValido(sudoku);

if(valido == 1){
return 0;


int situacao = situacaoSudoku(sudoku, vazias, valido);
free(vazias);
DesalocarSudoku(sudoku);
return situacao;
```

2 Impressões Gerais

A implementação do projeto foi bem pensada e projetada antes de começarmos o código, sentamos juntos para planejar o que queríamos programar e o que precisávamos para realizar a tarefa. Optamos por ter 3 Structs para facilitar o manuseamento das nossas variáveis e conseguir resolver com mais facilidade as questões a nós apresentadas. Além das funções que foram apresentadas pelo enunciado do TP, fizemos funções de alocação dinâmica e desalocação. Aspectos que gostamos foi a ideia do trabalho já que percebemos que poderíamos utilizar recursividade para resolver o sudoku, não utilizamos entretanto por falta de prática e por só perceber no fim do trabalho que poderiam ser feitas de outras formas as funções. Ambos os estudantes já conheciam o sudoku então não foram adquiridos nenhum conhecimento ambíguo, somente a prática da programação foi desenvolvida.

3 Análise

Podemos analisar a partir dos resultados que poderia ser feita uma seleção de saídas para resolver o sudoku. E além disso foi percebido por nós uma possibilidade de realizar funções recursivas para solucionar o sudoku

4 Conclusão

Concluindo o trabalho tivemos pouca dificuldade para implementar o código porém tivemos um grande problema com memory leak pois acessamos memória errada. Entretanto com ajuda do Valgrind (Software recomendado pelo professor) conseguimos identificar o erro e finalizamos o TP sem mais nenhum problema.