Relatório do Trabalho 03 - Parte 2 da disciplina SSC0601

Nomes:	NºUSP:
Gabriel Sotto Rodrigues	11800952
Otávio Terra Roque	11372845
Roberto Kazushi Yuuki Junior	11395815

Descrição da solução:

O tabuleiro é representado por uma matriz de caracteres e é iniciada com espaços. Uma estrutura com dois atributos, um caractere e um inteiro, identificam o jogador e se ele é controlado pelo computador ou pelo usuário.

As jogadas são divididas em dois tipos: jogador controlado pelo usuário e jogador controlado pelo computador. A primeira é realizada de forma simples, lendo as coordenadas da posição desejada na matriz e alocando o símbolo do jogador nessa posição, isso só ocorre se a posição estiver vazia. A jogada controlada pelo computador é aleatória, para isso armazena-se em um vetor as posições vazias e é selecionada, com o auxílio da função *rand,* uma delas.

Uma verificação de vencedor é feita a cada jogada, depende da última posição marcada e do tamanho da sequência mínima para vencer. Baseado nessas informações, o algoritmo verifica todas as possibilidades ao redor da última posição, isto é, nas linhas, colunas e diagonais nas quais ela está inserida.

Há duas variáveis que controlam cada partida: o número de jogadas e outra que indica se há um vencedor. Enquanto o números de jogadas for menor que a dimensão do tabuleiro ao quadrado e não houver um vencedor, serão executadas as funções de jogadas. Ao final imprime-se quem venceu ou se houve empate.

Descrição do código:

main.c:

Inclui-se a biblioteca *jogo.h* que contém as principais funções de inserção, verificação, impressão e de zerar o tabuleiro. Declara-se um estutura *jogador_t* que armazena o caractere do jogador em específico e um *int* que representa se ele é controlado ou não pelo computador.

Na função *main,* declara-se duas variáveis, *n* para o tamanho do tabuleiro e *reiniciar* que possibilita a escolha do usuário de jogar novamente com um auxílio de um *do while*. Neste, lê-se o tamanho do tabuleiro, armazenando-o em n, aloca-se dinamicamente um espaço correspondente à um vetor de ponteiros para ponteiros, preenchendo cada elemento do vetor

com um ponteiro para *char* que corresponde a uma matriz de *char n* por *n.* A função *zerar Tab* é chamada para iniciar essa matriz com espaços.

As variáveis *minSeq* e *m* são lidas e representam, respectivamente, a mínima sequência para a vitória e o número de jogadores. Baseado em *m,* aloca-se dinamicamente um vetor de ponteiros para *jogador_t* que armazenam os caracteres que representam cada jogador e o inteiro que permite identificar se o jogador é controlado ou joga automaticamente. Um *for* é iniciado para que o usuário insira o caractere e inteiro de cada jogador.

Declara-se *jogadas, vencedor, jogAt, ultimoX e ultimoY* que representam, respectivamente, a quantidade máxima de jogadas que será decrementada a cada jogada e que é essencial para caracterizar um empate, o índice do vencedor no vetor de jogadores, o índice do jogador atual no vetor de jogadores e, por fim, *ultimoX* e *ultimoY* representam as coordenadas na matriz da última posição escolhida pelos jogadores que são essenciais para a verificação de um vencedor.

O *while* declarado após essas variáveis é responsável por controlar as jogadas de todos os jogadores, é interrompido quando *jogadas* chega em zero ou *vencedor* possui um índice válido no vetor de jogadores. Dentro dessa estrutura de repetição, é imprimido na tela o tabuleiro por meio da função *rendertab* e, logo após, verifica se o jogador atual é controlado ou não por meio do inteiro presente no elemento correspondente do vetor de jogadores. Se o jogador atual for controlado, aciona-se a função *player* e verifica-se através da função *verificar* se há um vencedor, caso contrrário, a função *cpu* é acionada e a mesma verificação é feita. Armazena-se o valor do resto da divisão de *jogAt* incrementado pela quantidade de jogadores em *jogAt,* o que indica o próximo jogador mesmo chega na última posição do vetor de jogadores. Decrementa-se *jogadas* já que mais uma foi realizada.

Após a interrupção do *while,* aciona-se *rendertab* novamente para mostrar o tabuleiro final e também é verificado se houve empate ou vitória por meio de um *if* que verifica se o valor armazenado em *vencedor* corresponde ou não a uma posição no vetor de jogadores, imprime-se “Empate” se não ou “Jogador X venceu” se sim, onde ‘X’ é substituído pelo caractere do jogador vencedor. Libera-se a memória alocada para o tabuleiro e para o vetor de jogadores. Por fim, a variável *reiniciar* é atualizada de acordo com a entrada do usuário.

jogo.c:

A função *zerarTab* recebe como parâmentros o endereço da matriz de caracteres que representa o tabuleiro e *n.* Duas estruturas do tipo *for* são encadeadas para percorrer cada posição da matriz e alocar um espaço nelas.

A função *player* recebe o endereço do tabuleiro, o caractere a ser inserido e ponteiros para inteiros que são atualizados ao final dessa função para corresponder a última posição alocada. Declara-se dois inteiros *lin* e *col* que representam ,respectivamente, a linha e a coluna. Dentro de uma estrutura *do while,* são lidos *lin e col* digitados pelo usuário até que o mesmo digite uma posição vazia, prevenindo-se da possibilidade de sobrescrever uma posição já marcada. Por fim atualiza-se, os valores das coordenadas da última posição adicionada e o caractere correspondente é inserido nessa posição.

A função *cpu* recebe o endereço do tabuleiro, o tamanho do mesmo, o caractere do jogador controlado pelo computador, a quantidade de jogadas possíveis e a coordenadas da última posição atualizada, com o mesmo intuito da função *player.* Aloca-se memória dinamicamente para o vetor de ponteiros para char *vazias* e a matriz de inteiros *pos,* a primeira é utilizada para alocar o endereço de todas as posições vazias do tabuleiro para que uma delas seja escolhida aleatoriamente, a segunda armazena as coordenadas de cada um dos endereços armazenados em *vazias.* A variável *jogadas* é usada para para alocar a quantidade certa de memória, uma vez que a quantidade de posições vazias é numericamente igual a *jogadas.* Cria-se uma variável *aux,* iniciada com 0, e um duas estruturas do tipo *for* encadeadas para percorrer todo o tabuleiro com o intuito de armazenar em *vazias* e *pos* todas as informações das posições com espaço apenas. Logo após, com o auxílio da função *srand e rand,* armazena-se um valor aleatório no inteiro *selec* que é utilizado, em seguida, para determinar uma das posições vazias armazenadas em *vazias.* Atualiza-se o valor da última coordenada atualizada e armazena o caractere do jogador na posição correspondente da matriz. Ao fim da função, libera-se a memória utilizada por *vazias* e *pos.*

A função *verificar,* diferente das funções anteriores, retorna um inteiro que corresponde a um jogador do vetor de jogadores ou a uma posição inválida do mesmo que indica que ninguém ganhou até o determinado instante. Essa função baseia-se nas coordenadas da última posição atualizada e no tamanho mínimo de sequência para vencer, utilizando-as para restringir as posições a serem verificadas e recebe como parâmetros o tabuleiro, as coordenadas da última posição atualizada, o tamanho do tabuleiro, o tamanho de

sequência mínima, o caractere do jogador atual e seu índice no vetor de jogadores. As variáveis *lin e col* são iniciadas com 0 e representam a quantidade de posições em sequência na linha e coluna, respectivamente, que contenham a última posição atualizada. Para fazer a verificação nas direções vertical e horizontal, é utilizado uma estrutura *for* para percorrer essa linha da direita para a esquerda e essa coluna de cima para baixo, o valor de *lin* e de *col* são incrementados caso o caractere do jogador atual seja encontrado e zerados caso contrário. Após a execução deste *for* adquire-se a informação de que o jogador atual ganhou ou não pelas linhas ou colunas, a informação é testada ao comparar o valor de *lin e col* com o tamanho mínimo de sequência, retorna-se o índice do jogador caso algum deles seja maior. Caso isto não ocorra, verifica-se as diagonais, é analisado a parte superior em relação a última posição atualizada e a parte inferior tanto da diagonal "principal", quanto a "secundária". Essa verificação ocorre por meio de várias estruturas do tipo *while* que são encerrados ao encontrar a "borda" da matriz, a posição verificada não conter o caractere buscado ou já ter atingido o tamanho mínimo, dentro deles realizam-se incrementos ou decrementos, dependendo da matriz e qual parte está sendo analisada, nas coordenadas das posições a serem verificadas e incrementos em um inteiro, *soma,* que armazena o tamanho da sequência analisada até o momento. São várias estruturas desse tipo para que sejam analisadas as duas diagonais e as duas partes de cada uma delas. Por fim, verifica-se se há um vencedor ou não, retornando o índice correspondente, -1 ou o índice do jogador atual.

A função *rendertab* recebe o tabuleiro e o tamanho do mesmo. A primeira instrução é de limpar a linha de comandos, depois, com um *for,* imprime-se as coordenadas das colunas. Em seguida, através de duas estruturas de repetição do tipo *for* são encadeadas para imprimir na tela as coordenadas das linhas e as posições de cada coordenada do tabuleiro.

A função *freemat* recebe o tabuleiro e o seu tamanho, por meio de um *for* aplica-se a função *free* para cada ponteiro para char e depois para o vetor de ponteiros para ponteiro, o que libera a memória alocada para a representação do tabuleiro.

Instruções para a utilização:

1- Após utilizar o *make run,* digite o valor inteiro correspondente a dimensão do tabuleiro.

2- Digite o inteiro correspondente ao tamanho da sequência mínima para vencer.

3- Digite o número inteiro que corresponde a quantidade de jogadores desejado.

4- Digite o caractere que representará cada jogador, antes da inserção será informado o número desse jogador que corresponde a ordem de jogada também, isto é, jogador 1 joga primeiro e assim sucessivamente. Logo após a inserção do caractere, será questionado se esse jogador jogará automaticamente, digite ‘0’ para sim ‘1’ para não.

5- Aparecerá o tabuleiro atual, caso não haja jogadores controlados pelo usuário será apresentado o tabuleiro final. Caso contrário, aparecerá também o caractere correspondente ao jogador e será esperado a entrada do usuário.

6- Deve-se digitar primeiro o número inteiro correspondente a linha e, separado por um espaço, o correspondente a coluna. Para facilitar há pode-se basear nos índices colocados ao redor do tabuleiro para encontrar esses números. Exemplo: em um tabuleiro 3x3, caso deseja-se inserir no posição do meio, deve-se digitar ‘1 1’.

7- Os passos 5 e 6 se repetem até que haja um vencedor ou não haja nenhuma posição vazia. Aparecerá, caso haja um vencedor a mensagem contendo o caractere que venceu, e “Empate” se empatar.

8- Por fim, aparecerá a mensagem “Jogar novamente(‘S’ para sim)?”, caso deseje jogar novamente, digite ‘S’. Se não, digite qualquer outro caractere.

Link para o vídeo:

Seção de resolução:
https://drive.google.com/file/d/1M1bb3_n8-r8Iao2oYDckV4AF2wLCOBJ-/view?usp=sharing

Execução do programa:
https://drive.google.com/file/d/1T-peA-HwT70HSkYA8C55MyhDByb0B6lD/view?usp=sharing