Computação I

Campo Minado

Álvaro de Carvalho Alves - 118183211

Nívea Martins Gomes dos Santos - 118149403

Rafaella Figueira Busch - 118198999

23 de novembro de 2018

Introdução

O objetivo principal do jogo escolhido consiste em explorar um campo que, em pontos aleatoriamente determinados pelo jogo há minas, sem que elas sejam detonadas utilizando a habilidade de lógica e percepção do usuário. No jogo criado, o jogador tem acesso ao menu que contém opções como (1) **Jogar**, (2) **Recordes**, (3) **Instruções**, (4) **Créditos** e (5) **Sair do Jogo**.

1. Na opção **Jogar** *(figura 1)*, o usuário consegue **determinar o tamanho do tabuleiro** desejado, tendo **4 opções** (8x8; 10x14; 16x20 e Personalizado). Sendo que, ao escolher o Personalizado, ele consegue determinar a quantidade de minas, de linhas e colunas. Após a entrada, o jogo começa. Na tela de jogo, aparece o campo com as possíveis ações do jogador sendo elas (a) **Abrir Campo**, (b) **Fixar Bandeira**, (c) **Remover Bandeira** e (d) **Sair do Jogo**. Caso, após abrir todos os campos possíveis, o jogador não atingir nenhuma bomba, ele é considerado o vencedor, consegue ver o tempo decorrido e pode incluir seu nome na Seção de Recordes. *(figura 2)*

    a. Ao escolher **abrir o campo**, o usuário entra com a coordenada do campo desejado e o jogo retorna a quantidade de bombas existentes ao redor daquele campo ou os espaços vazios até o ponto em que há bombas próximas ou retornará as bombas, acarretando a perda do jogador.

    b. Quando o jogador tem a certeza da coordenada de alguma bomba, é possível **fixar uma bandeira** no local. *(figura 3)*

    c. Para a desistência da aplicação da bandeira nesse local, é possível **remover o item do campo**.

    d. Essa opção permite ao jogador **sair do jogo** no decorrer da partida.

2. Na Seção de **Recordes**, é possível ver os recordes salvos naquela máquina com a seguinte formatação “Nome Do Jogador: X min X seg em um campo XxX com X minas” ou a exclusão dos recordes existentes. *(figura 4)*

3. Na tela de **Instruções**, é exibido uma forma resumida das instruções para a melhor jogabilidade do usuário. O conteúdo são as escolhas de ações durante o jogo, a formatação para a escolha de campo e entre outros aspectos importantes. *(figura 5)*

4. Ao escolher a opção de **Créditos**, é exibida as informações dos criadores do jogo como Nome Completo e DRE.

5. A opção **Sair do Jogo** permite o usuário finalizar a execução do programa.

Metodologia

Foram utilizadas quatro bibliotecas no total. Além da stdio.h para as funções de entrada e saída, a biblioteca stdlib.h foi adicionada ao código para usar as funções de alocação dinâmica, a string.h para fazer comparações entre strings e a time.h para fazer a contagem do tempo levado para finalizar o jogo.

Funções de Entrada e Saída

Para dialogar com o usuário, seja dando instruções, mostrando as opções ou informando erros, foram utilizadas a função de saída **printf** durante todo o programa. A função contou com alguns argumentos no decorrer do código para colocar cores ao jogo utilizando %s e a Macro criada para armazenar a string da cor.

Para obter as escolhas do usuário como o campo escolhido, a ação e a opção desejada foram utilizadas as funções de entrada como a **scanf** e a **getchar**. Essas funções tiveram como parâmetros as variáveis que deveriam armazenar a resposta do jogador.

Além disso, há uma função especial que combina comandos de repetição e a printf para imprimir o tabuleiro do jogo de maneira organizada e com boa visualização no terminal do usuário.

Matrizes, Ponteiros e Alocação Dinâmica

Para representar o tabuleiro internamente, foi utilizado o conceito de matrizes. Há três matrizes no total, a matriz interna, a matriz do usuário e a matriz de status. A **matriz interna** guarda todos os campos do tabuleiro, ou seja, as posições das bombas, os números e os espaços vazios que são preparados aleatoriamente para cada jogo. A **matriz do usuário** guarda as partes da matriz interna que serão impressas no terminal, isto é, os campos que já foram abertos pelo usuário; Todas as operações disponíveis serão feitas nessa matriz. A **matriz de status** guarda se os campos foram abertos ou ainda estão fechados, ela é utilizada para validar a entrada do usuário e evitar que se faça operações com campos que já estão abertos.

A manipulação das matrizes foi feita através de **ponteiros**. A matriz de status é manipulada por um **ponteiro para inteiro**. A matriz interna e a do usuário são manipuladas por um **ponteiro para ponteiro para char**, isso é devido ao caractere especial utilizado para representar o campo fechado, que só pôde ser tratado como uma string.

```
char **matriz_interna, **matriz_usuario;
int *status_matriz;
```

Devido ao fato que há vários tamanhos de tabuleiro e, inclusive, a possibilidade de escolher um tamanho personalizado, as funções de **alocação dinâmica** foram utilizadas no jogo. Para alocar os espaços de memória, a *Calloc* foi chamada com os argumentos referentes ao tamanho das linhas e das colunas definidos anteriormente pelo usuário. Ao fim de cada partida a função *Free* é chamada para liberar todos os espaços alocados e evitar encher a memória do usuário.

```
matriz_interna = (char **) calloc( linhas * colunas, sizeof(char *) );
status_matriz = (int *) calloc( linhas * colunas, sizeof(int) );
matriz_usuario = (char **) calloc( linhas * colunas, sizeof(char *) );
```

Funções de teste e Arquivos

Para checar se o usuário inseriu a opção corretamente, foram usadas funções de teste. Essas funções testam as entradas do usuário e permitem a continuidade do programa — quebrando o while usado para repetir os testes — caso o resultado esperado ocorra, e pede novamente a entrada caso haja algum problema, exibindo uma mensagem de erro.

```
if ((sscanf(teste_entrada, "%d", &escolha_fim)) == 1 && (escolha_fim ==
1 || escolha_fim == 2)) break;
else printf(" \n Opção inválida. Digite novamente: \n");
```

Ao fim do jogo os dados de recorde dos jogadores são armazenados usando a ferramenta arquivo, com três funções para manipulação dos dados. A função salva_recorde salva o nome, tempo, tamanho do campo e número de minas do jogador dentro de um arquivo de texto, utilizando um ponteiro especial do tipo FILE para abrir um arquivo. A função print_recorde exibe esse arquivo txt com os dados do vencedores anteriores. A função apaga_recorde deleta esses dados e deixa o arquivo em branco.

```
void salva_recorde(char *nome, int tempo_min, int tempo_sec, int
qtd_minas, int linhas, int colunas);
void print_recorde();
void apaga_recorde();
```

Saídas das Execuções

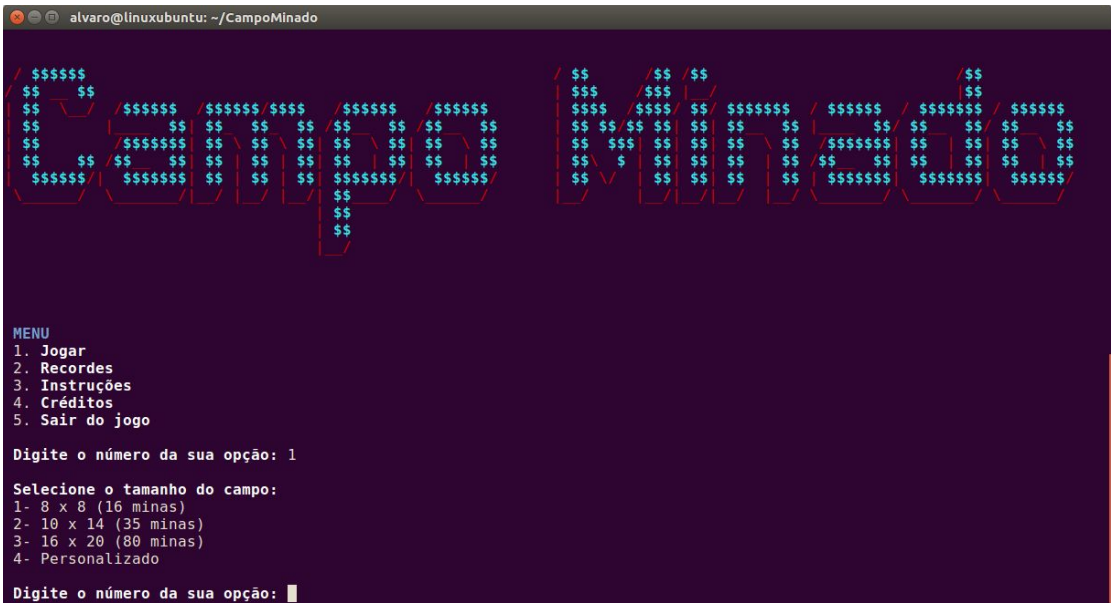

Figura 1 - Menu na opção Jogar

Figura 2 - Tela de Vitória

Figura 3 - Fixação da Bandeira

Figura 4 - Menu na opção Recordes

Figura 5 - Menu na opção Instruções

Conclusão

Após encontrar problemas quanto ao modo que poderiam ser aplicadas algumas funções cruciais para o funcionamento do programa, a necessidade de algumas pesquisas ficou exposta. O grupo pesquisou em sites, apostilas e vídeos e encontrou a existência de funções prontas em headers como a time.h para a elaboração do contador de tempo decorrido, meios de aplicar cor no terminal com as strings, etc.

Dessa forma, o comprometimento com uma boa execução do jogo, feito em um código limpo em C Ansi, levou o grupo a expandir a noção de programação, utilizando de todos os métodos aprendidos durante o curso de Computação I, além de outras informações decorrentes de pesquisas. Esse trabalho foi uma grande oportunidade para consolidar todos esses conceitos aprendidos e elevar o nível de raciocínio lógico para desenvolver um programa conciso que integre todas as funções desenvolvidas de maneira correta e limpa. Assim, fica clara a quantidade de experiência e conhecimento adquirido ao longo desse semestre.

Referências Bibliográficas

Kernigham, B. W., e Ritchie, D. M., The C Programming Language. 2. ed. Prentice Hall, New Jersey, 1988.

CRUZ, A. J. O. Curso de Linguagem C. Rio de Janeiro, 2016.