
Centro Federal de Educação Tecnológica de Minas Gerais

Informática _ INFO1

Trabalho de ALP/LALP

Relatório

Disciplina: ALP/LALP

Professor: Elizabeth S. Duane

---

Nomes: Felipe Augusto do Nascimento

Gabriel Rocha Viana

Luisa Eduarda Lemos Tibúrcio dos Santos

Izabela Rodrigues de Souza

Contagem

Novembro / 2020

# Explicação

Foram utilizadas 6 funções para a resolução desta atividade sendo elas:

```
1. int escolha (int i, int j, int X);
```

A função **escolha** recolhe os dados do usuário e efetua a parte mais importante do código que é mostra a tabela e a atualiza de acordo com os números escolhidos pelo usuário. Os parâmetros i, j e X é respectivamente: O valor da coluna, o valor da linha e o valor de quantas linhas e colunas tem a matriz (A variável X é preenchida antes da função ser iniciada pois se o jogador inicia o modo fácil X equivale a 10 e assim por diante).

Nesta função utilizamos o raciocínio de enquanto o valor da matriz não seja -1 (Bomba) ele irá repetir o código que forma a matriz (de maneira padrão) e dá as escolhas sobre ( Explorar que seleciona o local que o usuário deseja, Bandeira que coloca uma bandeira no local onde o usuário não tem certeza se tem bombas (Ao fazer isso é retirado 1 bomba do contador para “enganar” o jogador) e o de dúvida caso o jogador não tenha a menor ideia do que está por trás da posição). Logo após é estruturado uma matriz nova já com a escolha realizada. A função verifica se na casa equivale a -1 e caso seja a função irá mostrar os locais com as bombas e a frase “Você perdeu” e o tempo gasto (Utilizamos uma biblioteca time.h que tem essa função incluída).Se o valor não for -1 e sim a sua pontuação equivaler a 90, 185, 301(Multiplica o valor de casas por ele mesmo e subtrai o valor das casas, Ex: 10 * 10 = 100, 100 - 10 = 90). Caso a pontuação do jogador for igual a um desses critérios ele irá ganhar a partida.

```
2. int numero_bombas(int num_casas, int i, int j);
```

A função foi elaborada para verificar quantas bombas tem ao redor, a função verifica cada casa e se ela for -1(Bomba) ele adiciona +1 nas casas ao redor e caso tenha mais alguma bomba no redor ele adiciona +1 possibilitando a descoberta de quantas bombas tem por perto.

Na função nós verificamos as casas em volta utilizando os números inteiros i e j que foram declaradas nos roda_o_jogo a seguir, a variável num_casas é para verificar o tamanha da matriz para não deixar a função rodar fora do campo permitido. Foi utilizado os if para verificar cada casa ao redor que verifica se é o bomba ou não.

```
3. int roda_o_jogo_Facil(int num_Bomb, int num_casas);
```

Primeiramente o roda_o_jogo_Facil é uma estrutura básica que deu a origem ao **`roda_o_jogo_Medio`** e **`roda_o_jogo_Dificil`** então é quase igual o raciocínio das mesmas, primeiramente é criado a matriz com o código vazio(com o valor 10), logo em seguida ele inicia um while que enquanto o número de bombas (cont) for maior que a quantidade de casas da matriz fácil, dentro desse while ele gera 2 números aleatórios (a e b) que geram a quantidade num_casas (Ex: gera números de 0 a 10 no modo fácil), e depois junta esses dois valores para sortear onde a bomba vai ficar (O local sorteado irá receber o -1 e o cont vai ganhar +1) e isso irá acontecer até que o cont seja igual ao num_bomb (número de bombas (no caso do fácil = 10))

Em seguida o código gera a matriz tabela que antes de cada linha ele recebe um número para a identação das linhas (caso seja um número < 9 ele irá colocar um espaço a mais para ficar bonito), depois ele gera a matriz de caracteres que gera a matriz de '-' que não permite que o usuário veja a tabela de números inteiros.

Ao mesmo tempo que o usuário está vendo a matriz de caracteres o código está formando a tabela de números inteiros a partir da função **`numero_bombas`** caso o valor da casa já não seja uma bomba (-1).

Logo após ele recolhe os dados para a função escolha funcionar corretamente e logo após ele inicia a função escolha.

```
4. int roda_o_jogo_Medio(int num_Bomb, int num_casas);
```

Essa função teve como estrutura base a **roda_o_jogo_Facil.** Porém os valores foram alterados de acordo com os padrões da dificuldade.

Primeiramente foi criado a matriz com o código vazio (com o valor 10), logo em seguida ele inicia um while que enquanto o número de bombas (cont) for maior que a quantidade de casas da matriz médio, dentro desse while ele gera 2 números aleatórios (a e b) que geram a quantidade num_casas (Ex: gera números de 0 a 15 no modo médio), e depois junta esses dois valores para sortear onde a bomba vai ficar (O local sorteado irá

receber o -1 e o cont vai ganhar +1) e isso irá acontecer até que o cont seja igual ao num_bomb (número de bombas (no caso do médio = 40))

Em seguida o código gera a matriz tabela que antes de cada linha ele recebe um número para a identação das linhas (caso seja um número < 9 ele irá colocar um espaço a mais para ficar bonito), depois ele gera a matriz de caracteres que gera a matriz de '-' que não permite que o usuário veja a tabela de números inteiros.

Ao mesmo tempo que o usuário está vendo a matriz de caracteres o código está formando a tabela de números inteiros a partir da função **numero_bombas** caso o valor da casa já não seja uma bomba (-1).

Logo após ele recolhe os dados para a função `escolha` funcionar corretamente e logo após ele inicia a função `escolha`.

5. `int roda_o_jogo_Dificil(int num_Bomb, int num_casas);`

Essa função teve como estrutura base a **`roda_o_jogo_Facil`**. Porém os valores foram alterados de acordo com os padrões da dificuldade.

Primeiramente é criado a matriz com o código vazio(com o valor 10), logo em seguida ele inicia um while que enquanto o número de bombas (cont) for maior que a quantidade de casas da matriz Difícil, dentro desse while ele gera 2 números aleatórios (a e b) que geram a quantidade num_casas (Ex: gera números de 0 a 20 no modo Difícil ), e depois junta esses dois valores para sortear onde a bomba vai ficar (O local sorteado irá receber o -1 e o cont vai ganhar +1) e isso irá acontecer até que o cont seja igual ao num_bomb (número de bombas (no caso do Dificil = 99))

Em seguida o código gera a matriz tabela que antes de cada linha ele recebe um número para a identação das linhas (caso seja um número < 20 ele irá colocar um espaço a mais para ficar bonito), depois ele gera a matriz de caracteres que gera a matriz de '-' que não permite que o usuário veja a tabela de números inteiros.

6. `int fim(int i, int j, int num_casas, int U1, int U2);`

A função foi elaborada com o intuito de abrir as casas ao redor da casa aberta, porém esse código deu errado e não conseguimos fazê-lo rodar

por que o viana é um desprovido de inteligência e não conseguiu encontrar o erro no código. (ASS: Viana)

A função teve o mesmo raciocínio da função **numero_bombas** só que ao invés de verificar se vai ser -1 e contabilizar ele, ele deveria mostrar a casa caso não seja -1 mas não funcionou e nós não sabemos o motivo.

No **int main** foi declarado o início de todas as funções e procedimentos, sendo eles:

### 1. Tempo:

utilizando o complemento **srand(time(NULL));**, **time_t currentTime;**, **time(&currentTime);**,**typedef long time_t**; são utilizados como complementos do **include<time.h>**. Essas variáveis possibilita que eu consiga capturar os segundos, min e as horas só que no código só utilizamos os segundos. Para iniciar o cronômetro foi utilizado o código: **hora1 = time(NULL)**; que atribui o início da contagem a uma variável, para finalizar o cronômetro foi utilizado a função : **hora2 = time(NULL)**; que captura quanto tempo se passou e coloca em uma variável de controle. para mostrar ao usuário eu tive que colocar **hora2 - hora1** e atribuir a uma variável para retirar o tempo que estava sobrando.

### 2. Switch:

Foi utilizado para perguntar ao usuário qual função ele gostaria de executar, no primeiro **switch** perguntamos ao usuário se ele gostaria de joga (inicia o modo fácil), personalizado (pergunta qual dos três modos que ele gostaria de jogar) e caso ele escolha sair o código simplesmente fecha. Ao escolher jogar e personalizado o código pergunta o nome do jogador e formata a primeira tabela e a sua indentação para facilitar que o usuário

saiba a coluna que ele está mexendo , logo após ele inicia a função de seu respectivo modo de jogo e mostra o tempo que ele demorou para finalizar o jogo.

3. **`for:`**

   Foi utilizado para gerar o embelezamento da matriz colocando os números, traços (‘ _ ’ e ‘ - ’).

Muito obrigado Beth pelos seus serviços, foi um prazer imenso estar na mesma turma que a senhora está ministrando.