

INSTITUTO FEDERAL DE SANTA CATARINA - IFSC  
CURSO DE ENGENHARIA ELÉTRICA

ALUNO  
ELIAS ANZINI JUNIOR

PROFESSOR  
NIVALDO T. JUNIOR

**PROJETO DE PROGRAMAÇÃO II**

JOINVILLE, 2021

## SUMÁRIO

1. PROGRAMAÇÃO I.....	3
1.1. INTRODUÇÃO .....	3
1.2. TELAS .....	3
1.3. CÓDIGO .....	5
2. PROGRAMAÇÃO II .....	12
2.1. ARQUIVOS NECESSÁRIOS .....	12
2.2. TELAS .....	13
2.3. FUNÇÕES ADICIONADAS .....	16
2.2.1. CRIAÇÃO DE TELA ATRAVÉS DE .TXT .....	16
2.2.2. RANKING .....	18
2.4. CÓDIGO .....	21

# 1. PROGRAMAÇÃO I

## 1.1. INTRODUÇÃO

Este projeto consiste no desenvolvimento de um jogo estilo *playfield* utilizando código C. O programa deve se baseado em um labirinto com desafios crescentes. O conteúdo abaixo é referente ao jogo desenvolvido em programação I, as implementações desenvolvidas podem ser visualização no tópico PROGRAMAÇÃO II.

## 1.2. TELAS



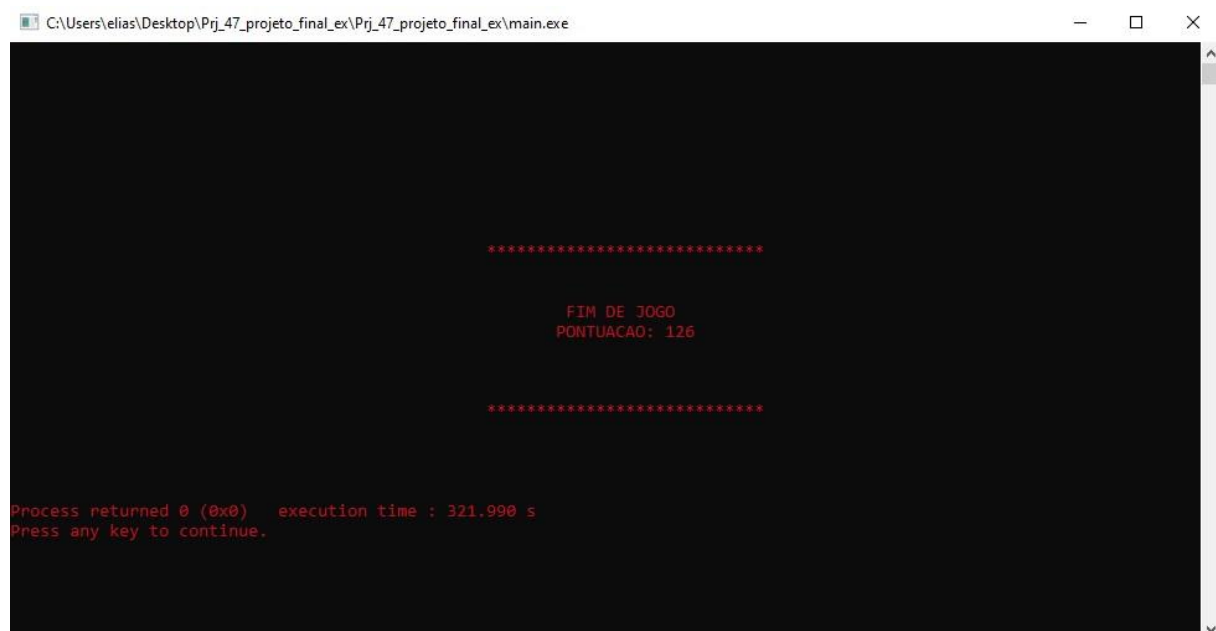
Figura 1: Primeira Fase




Figura 2: Segunda Fase



**Figura 3:** Terceira Fase



**Figura 4:** Tela Final

 Passagem Secreta

### 1.3. CÓDIGO

Abaixo segue alguns comentários sobre as principais partes do código. Foram utilizadas partes do primeiro código, o restante segue os mesmos padrões.

[illegible]

**Figura 5: Matrizes**

Para o desenvolvimento das matrizes, foi utilizado somente o símbolo 178 do código ASCII para a criação do mapa. Através disso torna-se mais fácil a criação das restrições para passagem da parede. Os obstáculos utilizados foram o código 254 da tabela ASCII.

```
for(i = 0; i < linhas; i++)
{
    for(j = 0; j < colunas; j++)
    {
        gotoxy(j + posx_inicial , i + posy_inicial );
        printf("%c", tela_l[i][j]);
    }
}
```

**Figura 6:** Plotamento do mapa

Para plotar o mapa, foi utilizado o código fornecido pelo professor.

Utilizando KEY == 1 como exemplo, foi utilizado um if para restringir a passagem nos blocos 178 e 254. Essa passagem só não ocorre quando for uma coordenada de um caminho secreto ou de um bloco 254 quebrado.

```

gotoxy(89,9);
if(KEY == 1)
{
    seta = 1;
    printf("Seta para cima");
    y--;
    gotoxy(x, y+1);
    printf("%c",255);
    //*****
    // Restrições de passagem
    //*****
    if (x!=30 || y!=28) // Passagem aceita somente quando x == 30 e y ==28
    {
        if ((tela_1[y-desl_y + 1][x-desl_x + 1] == 178) || (tela_1[y-desl_y + 1][x-desl_x + 1] == 254))
        {
            gotoxy(60,17); // texto informativo só para saber se encontrou uma barreira
            y++;
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont); //Contador de Pontua
        }
        else //Contador de Pontua
        {
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont+1);
            cont++;
        }
    }
    else // Se for parede secreta ele ganha mais 5 pontos na próxima vez que passa nela, se não for parede secreta a pontuação da pessoa normalmente
    {
        gotoxy(90,7);
        printf("Pontuacao = %d ", cont+1);
        cont++;
        if (parede == 0)
        {
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont - 5);
            cont -= 5;
            parede++;
        }
    }
    //*****
    if(y < lim_y_top)
        y = lim_y_top;
}

```

**Figura 7:** Restrição de passagem

Cada bloco 254 tem sua própria variável (parede1, parede2, parede3, etc), quando ele é quebrado seu valor é definido como '1', nesse momento a passagem pelo bloco não é mais restringida pelo if. Isso pode ser visto mais claramente através da KEY == 2, na imagem abaixo.

```

else if(KEY == 2)
{
    printf("Seta para baixo");
    y++;
    gotoxy(x, y-1);
    printf("%c",255);
    seta = 2;
    //*****
    // Restrições de passagem
    //*****
    if ((x!=30 || y!=28) && (((x != 9 || y != 10) || (parede2 == 0)))) //Se não for parede secreta
    {
        if (((tela_1[y-desl_y + 1][x-desl_x + 1] == 178) || (tela_1[y-desl_y + 1][x-desl_x + 1] == 254))
        {
            gotoxy(60,17);
            y--;
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont);
        }
        else
        {
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont+1);
            cont++;
        }
    }
    if ((parede2 == 1) && (x==9 && y==10))
    {
        gotoxy(90,7);
        printf("Pontuacao = %d ", cont+1);
        cont++;
    }
    if (x==30 && y==28) // Se for parede secreta
    {
        gotoxy(90,7);
        printf("Pontuacao = %d ", cont+1);
        cont++;
        if (parede == 0)
        {

```

**Figura 8:** Restrição de passagem

A restrição de passagem foi feita para todas as teclas de movimento.

```
if(KEY == 1)
{
    seta = 1;
    printf("Seta para cima");
    y--;
    gotoxy(x, y+1);
    printf("%c", 255);
    //*****
    // Restrições de passagem
    //*****
    if (x!=30 || y!=28) // Passagem somente quando x == 30 e y ==28;
    {
        if((tela_1[y-desl_y + 1][x-desl_x + 1] == 178) || (tela_1[y-desl_y + 1][x-desl_x + 1] == 254))
        {
            #define desl_x 7

            gotoxy(60,17); // texto informativo só para saber se encontrou uma barreira
            y++;
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont); //Contador de Pontos
        }
        else //Contador de Pontos
        {
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont+1);
            cont++;
        }
    }
}
else // Se for parede esquerda ela ganha mais 5 pontos na primeira vez que passa pela, as outras vezes a pontuação de passas normalmente
{
    gotoxy(90,7);
    printf("Pontuacao = %d ", cont+1);
    cont++;
    if (parede == 0)
    {
        gotoxy(90,7);
    }
}
```

Figura 9: Key == 1

```
else if(KEY == 2)
{
    printf("Seta para baixo");
    y++;
    gotoxy(x, y-1);
    printf("%c", 255);
    seta = 2;
    //*****
    // Restrições de passagem
    //*****
    if ((x!=30 || y!=28) && (((x != 9 || y != 10) || (parede2 == 0)))) //Se não for parede esquerda
    {
        if((tela_1[y-desl_y + 1][x-desl_x + 1] == 178) || (tela_1[y-desl_y + 1][x-desl_x + 1] == 254))
        {
            gotoxy(60,17);
            y--;
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont);
        }
        else
        {
            gotoxy(90,7);
            printf("Pontuacao = %d ", cont+1);
            cont++;
        }
    }
}
if ((parede2 == 1) && (x==9 && y==10))
{
    gotoxy(90,7);
    printf("Pontuacao = %d ", cont+1);
    cont++;
}
if (x==30 && y==28) // Se for parede esquerda
{
    gotoxy(90,7);
    printf("Pontuacao = %d ", cont+1);
    cont++;
    if (parede == 0)
    {
        gotoxy(90,7);
        printf("Pontuacao = %d ", cont - 5);
        cont -= 5;
        parede++;
    }
}
```

Figura 10: Key == 2

```

else if(KEY == 3)
{
    printf("Seta para esquerda");
    x--;
    gotoxy(x+1, y);
    printf("%c",255);
    seta = 3;

    //*****
    // Restrições de passagem
    //*****
    if (((x != 13 || y != 11) || (paredel == 0)) && ((x != 13 || y != 22) || (pared3 == 0)))
    {
        if((tela_1[y-desl_y + 1][x-desl_x + 1] == 178) || (tela_1[y-desl_y + 1][x-desl_x + 1] == 254))
        {
            gotoxy(60,17);
            x++;
            gotoxy(90,7);
            printf("Fontuacao = %d ", cont);
        }
        else
        {
            gotoxy(90,7);
            printf("Fontuacao = %d ", cont+1);
            cont++;
        }
    }
    if ((paredel == 1) && (x==13 && y==11))
    {
        gotoxy(90,7);
        printf("Fontuacao = %d ", cont+1);
        cont++;
    }
    if ((pared3 == 1) && (x==13 && y==22))
    {
        gotoxy(90,7);
        printf("Fontuacao = %d ", cont+1);
        cont++;
    }
    //*****
    if(x < lim_x_left)
        x = lim_x_left;
}

```

Figura 11: Key ==3

```

else if(KEY == 4)
{
    printf("Seta para direita");
    x++;
    gotoxy(x-1, y);
    printf("%c",255);
    seta = 4;

    //*****
    // Restrições de passagem
    //*****
    if (((x != 13 || y != 11) || (paredel == 0)) && ((x != 13 || y != 22) || (pared3 == 0)))
    {
        if((tela_1[y-desl_y + 1][x-desl_x + 1] == 178) || (tela_1[y-desl_y + 1][x-desl_x + 1] == 254))
        {
            gotoxy(60,17);
            x--;
            gotoxy(90,7);
            printf("Fontuacao = %d ", cont);
        }
        else
        {
            gotoxy(90, 7);
            printf("Fontuacao = %d ", cont+1);
            cont++;
        }
    }
    if ((paredel == 1) && (x==13 && y==11)) //Se estiver em cima da parede 1
    {
        gotoxy(90,7);
        printf("Fontuacao = %d ", cont+1);
        cont++;
    }
    if ((pared3 == 1) && (x==13 && y==22)) //Se estiver em cima da parede 3
    {
        gotoxy(90,7);
        printf("Fontuacao = %d ", cont+1);
        cont++;
    }

    if(tela_1[y-desl_y + 1][x-desl_x + 1] == 178)
    {
        gotoxy(60,17);
        x--;
    }
    //*****
    if(x > lim_x_right)
        x = lim_x_right;
}

```

Figura 12: Key ==3



É possível observar no código a implementação da pontuação, onde foi utilizado o contador ‘cont’. É importante notar que, as passagens secretas e blocos quebrados pulavam a restrição, sendo necessário um contador independente para eles.

No labirinto, o personagem deve atirar com uma arma apertando a barra, isso pode ser visto na imagem abaixo, na Key == 5;

```
while (d<3)
{
    if (seta == 4)
    {
        if(((y==11) && (x==12)) && (paredel == 0)) || (((y==11) && (x==11)) && (paredel == 0))) //Posições que vai quebrar a parede = continua
        {
            paredel = 1;
            gotoxy(90, 7);
            printf("Pontuacao = %d ", cont-5);
            cont-=5;
        }
        if((((y==22) && (x==12)) && (pared3 == 0)) || (((y==22) && (x==11)) && (pared3 == 0)) || (((y==22) && (x==10)) && (pared3 == 0))) //Posições que vai quebrar a parede = continua
        {
            pared3 = 1;
            gotoxy(90, 7);
            printf("Pontuacao = %d ", cont-5);
            cont-=5;
        }
        if (tela_l[y-desl_y + 1][x-desl_x + d + 2] == 178)
        {
            break;
        }
        gotoxy(x+d+1, y);
        printf("%c", 250);
        Sleep(20);
        gotoxy(x+d+1, y);
        printf("%c", 255);
        d++;
    }
    if (seta == 2)
    {
        if(((y==8) && (x==9)) && (pared2 == 0)) || (((y==9) && (x==9)) && (pared2 == 0))) //Posições que vai quebrar a parede = continua
        {
            pared2 = 1;
            gotoxy(90, 7);
            printf("Pontuacao = %d ", cont-5);
            cont-=5;
        }
        if (tela_l[y-desl_y + 2 + d][x-desl_x + 1] == 178)
        {
            break;
        }
        gotoxy(x, y+d+1);
        printf("%c", 250);
        Sleep(20);
        gotoxy(x, y+d+1);
        printf("%c", 255);
        d++;
    }
}
```

**Figura 12:** Key ==5

Para simular os tiros foram utilizados pontos. A direção do tiro é definida pela variável ‘seta’. Foi utilizado um contador para printar a sequência de pontos e deletes para simular o tiro. Cada direção do tiro possui um if, que impede que blocos com código 178 sejam quebrados.

Como pode ser visto na próxima imagem, é necessário definir as posições do personagem que ele pode quebrar o bloco 254. Quando o espaço for selecionado e o personagem estiver nessa posição, o valor da parede é definida como 1; não sendo mais restringida na passagem.

```
if (seta == 2)
{
    if(((y==8) && (x==9)) && (pared2 == 0)) || (((y==9) && (x==9)) && (pared2 == 0))) //Posições que vai quebrar a parede = continua
    {
        pared2 = 1;
        gotoxy(90, 7);
        printf("Pontuacao = %d ", cont-5);
        cont-=5;
    }
    if (tela_l[y-desl_y + 2 + d][x-desl_x + 1] == 178)
    {
        break;
    }
    gotoxy(x, y+d+1);
    printf("%c", 250);
    Sleep(20);
    gotoxy(x, y+d+1);
    printf("%c", 255);
    d++;
}
```

**Figura 13:** Coordenadas para quebrar o alvo

```

while (d<3)
{
    if (seta == 4)
    {
        if((((y==11) && (x==12)) && (paredel == 0)) || (((y==11) && (x==11)) && (paredel == 0))) //Ensiãas que vai quebrar a parede = continuar
        {
            paredel = 1;
            gotoxy(90, 7);
            printf("RONTUACAAQ = %d ", cont-5);
            cont-=5;
        }
        if((((y==22) && (x==12)) && (paredel == 0)) || (((y==22) && (x==11)) && (paredel == 0)) || (((y==22) && (x==10)) && (paredel == 0))) //Ensiãas que vai quebrar a parede = continuar
        {
            paredel = 1;
            gotoxy(90, 7);
            printf("RONTUACAAQ = %d ", cont-5);
            cont-=5;
        }
        if (tela_l[y-desl_y + 1][x-desl_x + d + 2] == 178)
        {
            break;
        }
        gotoxy(x+d+1, y);
        printf("%c", 250);
        Sleep(20);
        gotoxy(x+d+1, y);
        printf("%c", 255);
        d++;
    }
    if (seta == 2)
    {
        if((((y==9) && (x==9)) && (paredel == 0)) || (((y==9) && (x==9)) && (paredel == 0))) //Ensiãas que vai quebrar a parede = continuar
        {
            paredel = 1;
            gotoxy(90, 7);
            printf("RONTUACAAQ = %d ", cont-5);
            cont-=5;
        }
        if (tela_l[y-desl_y + 2 + d][x-desl_x + 1] == 178)
        {
            break;
        }
        gotoxy(x, y+d+1);
        printf("%c", 250);
        Sleep(20);
        gotoxy(x, y+d+1);
        printf("%c", 255);
        d++;
    }
}

```

Figura 14: Key = 5

```

if (seta == 3)
{
    if (((y==11) && (x==16)) && (paredel == 0)) || (((y==11) && (x==15)) && (paredel == 0)) || (((y==11) && (x==14)) && (paredel == 0))
    {
        paredel = 1;
        gotoxy(90, 7);
        printf("RONTUACAAQ = %d ", cont-5);
        cont-=5;
    }
    if (((y==22) && (x==16)) && (paredel == 0)) || (((y==22) && (x==15)) && (paredel == 0)) || (((y==22) && (x==14)) && (paredel == 0))
    {
        paredel = 1;
        gotoxy(90, 7);
        printf("RONTUACAAQ = %d ", cont-5);
        cont-=5;
    }
    if (tela_l[y-desl_y + 1][x-desl_x - d] == 178)
    {
        break;
    }
    gotoxy(x-d-1, y);
    printf("%c", 250);
    Sleep(20);
    gotoxy(x-d-1, y);
    printf("%c", 255);
    d++;
}
if (seta == 1)
{
    if (tela_l[y-desl_y - d][x-desl_x + 1] == 178)
    {
        break;
    }
    gotoxy(x, y-d-1);
    printf("%c", 250);
    Sleep(20);
    gotoxy(x, y - d - 1);
    printf("%c", 255);
    d++;
}
}
d=0;
}

```

Figura 15: Key = 5

Para realizar a troca de telas, foi definida uma posição no mapa que quando atingido, ocorre o break do while, e a tela é limpada pelo comando clrscr(). Depois disso, é realizada a plotagem da próxima matriz.

```
else if (x == 36 && y == 31)
{
    clrscr();
    break;
}
```

**Figura 16:** Troca de Tela











Todas as outras telas seguem o padrão de códigos utilizados anteriormente.

## 2. PROGRAMAÇÃO II

### 2.1. ARQUIVOS NECESSÁRIOS

Os arquivos necessários para o jogo são o ProjetoPrincipal.c, arquivos de funções funções.c e funções.h. Além desses, o jogo importa as matrizes contidas nos arquivos de matriz.txt para criação das telas.

Quando um jogador finaliza o jogo e insere seu nome, é criado um arquivo chamado estrutura que armazena o nome e pontuação dos jogadores.

 funcoes.c	15/08/2021 13:06	Arquivo C	2 KB
 funcoes.h	15/08/2021 13:07	Arquivo H	1 KB
 Matriz1	08/08/2021 17:07	Documento de Te...	6 KB
 Matriz2	08/08/2021 18:00	Documento de Te...	6 KB
 Matriz3	08/08/2021 18:02	Documento de Te...	6 KB
 Matriz4	14/08/2021 20:55	Documento de Te...	8 KB
 Matriz5	14/08/2021 23:30	Documento de Te...	8 KB
 ProjetoPrincipal.c	15/08/2021 13:14	Arquivo C	76 KB
 my_lib.h	03/03/2021 18:49	Arquivo H	4 KB
 le_tecclas.h	06/03/2021 11:50	Arquivo H	2 KB

**Figura 16:** Arquivos necessários

## 2.2. TELAS

Foram adicionadas duas novas telas, além disso agora todas as fases possuem cores que indicam a dificuldade.

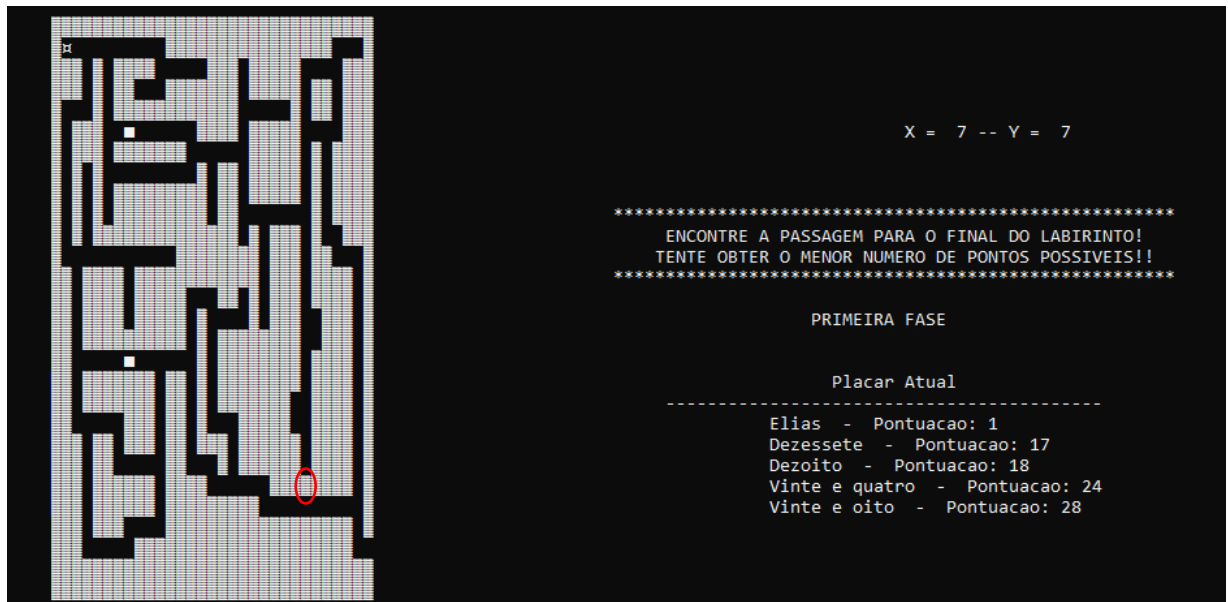


Figura 17: Tela 1



Figura 18: Tela 2



Figura 19: Tela 3




Figura 20: Tela 4



Figura 21: Tela 5



Figura 22: Tela 6

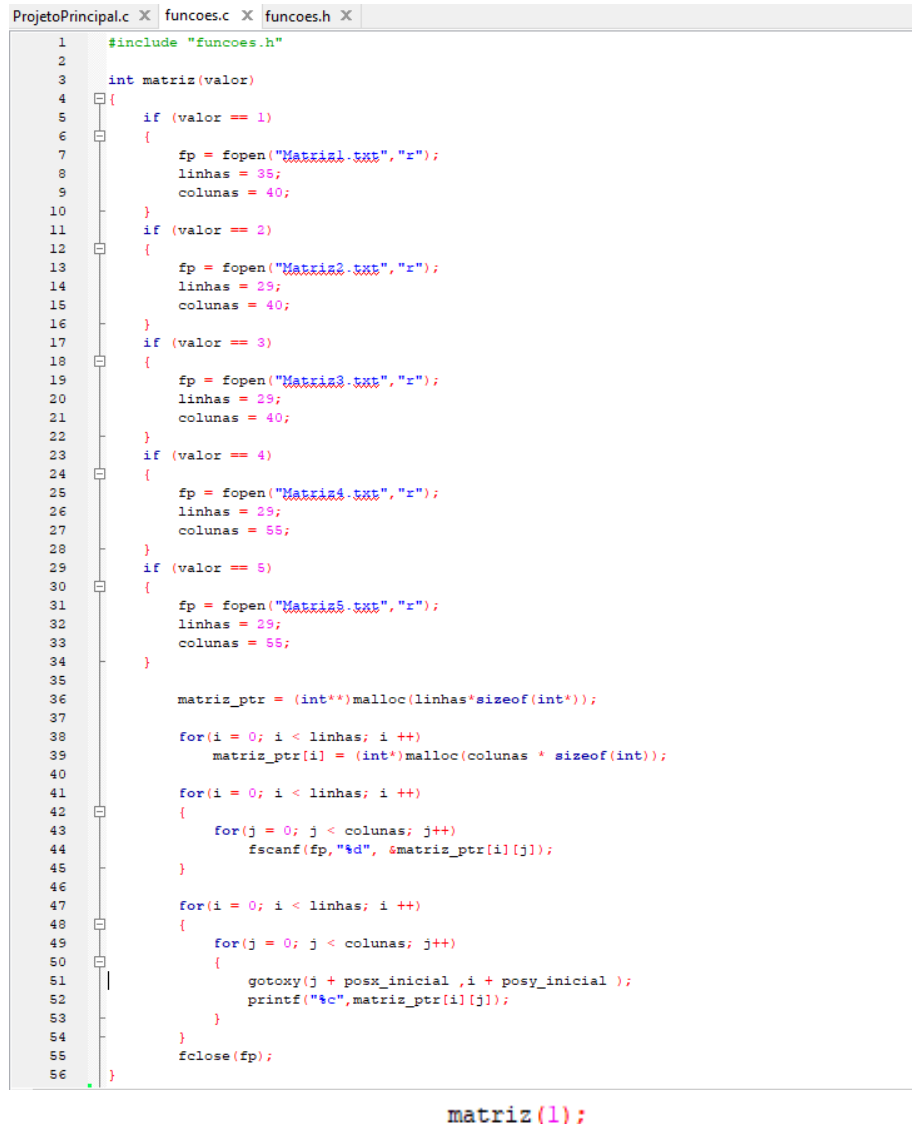
 Passagem Secretas

## 2.3. FUNÇÕES ADICIONADAS

### 2.2.1. CRIAÇÃO DE TELA ATRAVÉS DE .TXT

As telas agora são importadas dos arquivos de texto, desta vez é utilizada somente um vetor para armazenar a matriz da tela, como pode ser visto no código abaixo é a `matriz_ptr`. No final de cada tela é realizado o comando `free(matriz_ptr)` para liberar a memória alocada para armazenamento do vetor. Para identificar qual matriz será importada na função é utilizado o parâmetro `valor`.

Além disso, foi corrigido o bug de posicionamento da tela. Quando cada matriz é importada é definido o número de linhas e colunas ideais para ela, de acordo com o utilizado na criação do arquivo txt.



```
ProjetoPrincipal.c X funcoes.c X funcoes.h X
1  #include "funcoes.h"
2
3  int matriz(valor)
4  {
5      if (valor == 1)
6      {
7          fp = fopen("Matriz1.txt", "r");
8          linhas = 35;
9          colunas = 40;
10     }
11     if (valor == 2)
12     {
13         fp = fopen("Matriz2.txt", "r");
14         linhas = 29;
15         colunas = 40;
16     }
17     if (valor == 3)
18     {
19         fp = fopen("Matriz3.txt", "r");
20         linhas = 29;
21         colunas = 40;
22     }
23     if (valor == 4)
24     {
25         fp = fopen("Matriz4.txt", "r");
26         linhas = 29;
27         colunas = 55;
28     }
29     if (valor == 5)
30     {
31         fp = fopen("Matriz5.txt", "r");
32         linhas = 29;
33         colunas = 55;
34     }
35
36     matriz_ptr = (int**)malloc(linhas*sizeof(int*));
37
38     for(i = 0; i < linhas; i++)
39         matriz_ptr[i] = (int*)malloc(colunas * sizeof(int));
40
41     for(i = 0; i < linhas; i++)
42     {
43         for(j = 0; j < colunas; j++)
44             fscanf(fp, "%d", &matriz_ptr[i][j]);
45     }
46
47     for(i = 0; i < linhas; i++)
48     {
49         for(j = 0; j < colunas; j++)
50         {
51             gotoxy(j + posx_inicial, i + posy_inicial);
52             printf("%c", matriz_ptr[i][j]);
53         }
54     }
55     fclose(fp);
56 }
```

`matriz(1);`

Figura 19: Função de criação de tela `int matriz(valor)`





### 2.2.2. RANKING

Um placar foi criado para exibir os melhores jogadores da partida (aqueles que fazem menos pontos). Após o final da partida o usuário deve preencher seu nome e o placar final com os seis melhores é exibido. Durante todas as fases o placar é exibido na tela.

```
printf("Digite o nome: ");
gets(lista[0].jogador);
lista[0].pontuacao = cont;
fflush(stdin);

// Cola a estrutura no txt
placar = fopen("Estrutura.txt", "a");
fprintf(placar, "%s - Pontuacao: %d\n", lista[0].jogador, lista[0].pontuacao);
fclose(placar);
```

**Figura 22:** Código que passa os dados do jogador para o .Txt

Como pode ser visto no código acima, os dados dos usuários são passados para o arquivo .txt através do comando fprintf. É utilizado o modo “a” para que os dados do jogador sempre sejam escritos em sequência no arquivo. O nome do arquivo criado com todos os dados dos jogadores é Estrutura.txt.

```
placar = fopen("Estrutura.txt", "r");

// Passa todo o txt para a estrutura

for (k = 0; k < 100; k++)
{
    fscanf(placar, "%[^\\-] - Pontuacao: %d\\n", &lista[k].jogador, &lista[k].pontuacao);
}
printf("\\n\\n\\n");
fclose(placar);
```

**Figura 23:** Código que passa os dados do arquivo para a estrutura

Na imagem acima, é exibido o código que realiza a leitura de todo o arquivo texto e passa para a estrutura chamada Jogadores, é utilizado o fscanf para isso.

```

struct jogadores
{
    char jogador[50];
    int pontuacao;
} lista[100];

struct jogadores *ptr;
ptr = &lista;
struct jogadores aux;

```

**Figura 24:** Estrutura com os membros e ponteiro criado

```

for (k = 0; k < 100; k++)
{
    for (j = 0; j < 100; j++)
    {
        if ((lista[k].pontuacao < lista[j].pontuacao) && (lista[k].pontuacao != NULL))
        {
            maior = lista[k].pontuacao;
            aux = ptr[k];

            lista[k].pontuacao = lista[j].pontuacao;
            ptr[k] = ptr[j];

            lista[j].pontuacao = maior;
            ptr[j] = aux;
        }
    }
}

```

**Figura 25:** Comando de Lógica

Após todos os usuários do arquivo txt serem armazenados na estrutura é realizado o comando de lógica para posicionar os melhores jogadores na tabela. Como pode ser visto, apenas os 100 primeiros usuários são ordenados, o mesmo número utilizado para criação da estrutura.

```

}
printf("\t\t\t\t\tPlacar\n");
printf("\t\t\t\t\t-----");
printf("\n");

// Printa os valores da estrutura
for (k = 0; k < 6; k++)
{
    if (lista[k].pontuacao != NULL)
        printf("\t\t\t\t\t%s - Pontuacao: %d\n", lista[k].jogador, lista[k].pontuacao);
}
fclose(placar);

return 0;
}

```

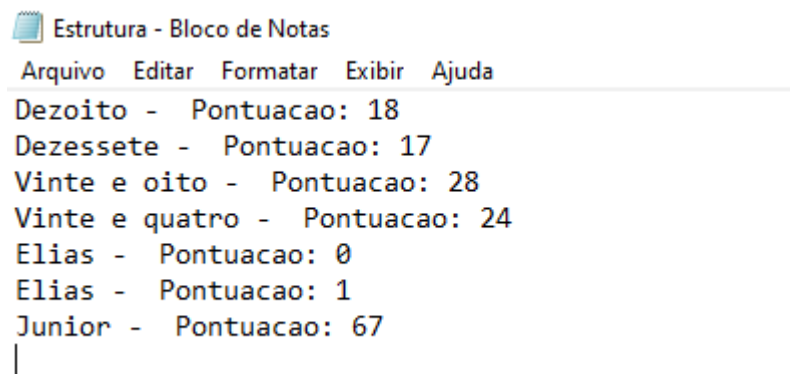
**Figura 26:** Comando para exibir o placar ordenado

Após a estrutura ser organizada, são printados os cinco primeiros usuários da tabela.

Para exibir o placar em todas as telas, é necessário em todas elas colocar os comandos de leitura do arquivo, organização da estrutura e print da estrutura.

```
printf("PRIMEIRA FASE");  
  
//***** PLACAR *****  
//*****  
  
placar = fopen("Estrutura.txt","r");  
  
for (k = 0; k < 100; k++)  
{  
    fscanf(placar,"%[^-] - Pontuacao: %d\n",&lista[k].jogador,&lista[k].pontuacao);  
}  
  
printf("\n\n\n");  
fclose(placar);  
  
struct jogadores *ptr;  
ptr = &lista;  
struct jogadores aux;  
  
for (k = 0; k < 100; k++)  
{  
  
    for (j = 0; j < 100; j++)  
    {  
        if ((lista[k].pontuacao < lista[j].pontuacao) && (lista[k].pontuacao != NULL))  
        {  
            maior = lista[k].pontuacao;  
            aux = ptr[k];  
  
            lista[k].pontuacao = lista[j].pontuacao;  
            ptr[k] = ptr[j];  
  
            lista[j].pontuacao = maior;  
            ptr[j] = aux;  
        }  
    }  
}  
  
printf("\t\t\t\t\t\t\t\t\t\t\t Placar Atual\n");  
printf("\t\t\t\t\t\t\t\t\t\t\t ----- \n");  
  
for (k = 0; k < 6; k++)  
{  
    if (lista[k].pontuacao != NULL)  
        printf("\t\t\t\t\t\t\t\t\t\t\t %s - Pontuacao: %d\n",lista[k].jogador, lista[k].pontuacao);  
}
```

**Figura 27:** Exemplo primeira fase placar atual



**Figura 28:** Bloco de notas com os usuários criados

## **2.4. CÓDIGO**

Para criação das novas telas, foram utilizados os mesmos princípios da Programação I, por esse motivo o código das novas telas não está detalhado no relatório.