

## **Trabalho Prático 0**

Matheus Nascimento Peixoto - 4662

Florestal - MG  
2023

# Sumário

|                                  |          |
|----------------------------------|----------|
| <b>Introdução</b>                | <b>3</b> |
| <b>Pré-requisitos e execução</b> | <b>3</b> |
| <b>Metodologia</b>               | <b>3</b> |
| <b>Desenvolvimento</b>           | <b>4</b> |
| “desenhaBordas.c”                | 4        |
| “simbolos.c”                     | 4        |
| <b>Resultados</b>                | <b>6</b> |
| <b>Conclusão</b>                 | <b>7</b> |
| <b>Referências</b>               | <b>8</b> |

## Introdução

Neste trabalho o objetivo foi criar desenhos, dentro de um quadro 20x80, de maneira aleatória. Os desenhos foram um asterisco simples, um sinal de adição e uma letra 'X', ambos também formados por asteriscos.

Além dessas três, foi solicitada a criação de um quarto desenho, à escolha do aluno, sendo realizado um quatro (4) dentro de uma caixa.

O resultado final do projeto pode ser encontrado no repositório do GitHub disponível em [1].

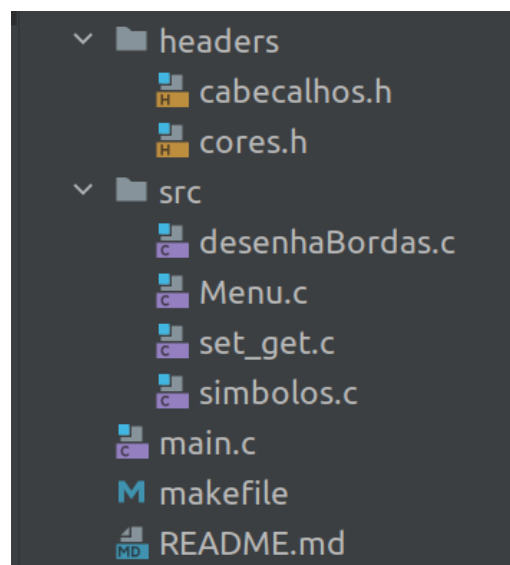
## Pré-requisitos e execução

O trabalho foi desenvolvido em C e no Sistema Operacional Linux. Para compilar e executar o projeto é necessário ter GCC e make instalado no sistema. Para compilar, basta digitar **make all** e em seguida **make run** no terminal.

## Metodologia

Durante a realização deste TP foi utilizado o GitHub, para o versionamento, e foi este foi organizado em pastas, com a “headers” contendo os cabeçalhos da funções e as cores utilizadas e a pasta “src”, contendo os códigos das funções utilizadas, juntamente com o “menu”, o qual foi criado visando deixar o código usuário (main.c) ficar mais limpo.

A seguir, na **Figura 1** é possível observar tal organização.



**Figura 1 - Organização do projeto**

## Desenvolvimento

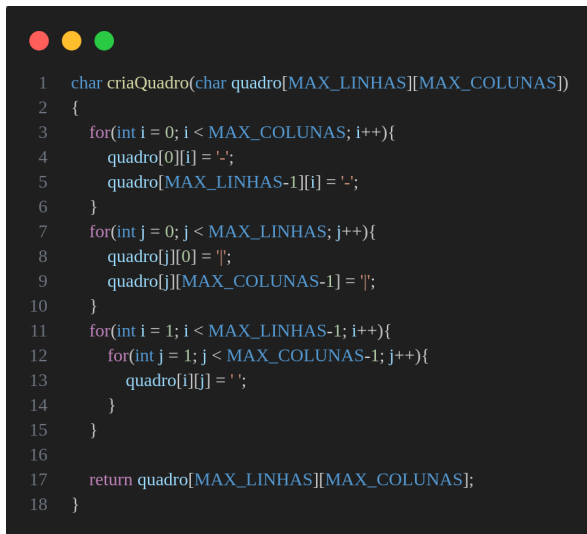
Nos subcapítulos a seguir serão apresentados os arquivos essenciais deste trabalho e explicado o funcionamento de suas principais funções.

### “desenhaBordas.c”

Nesse arquivo estão as funções responsáveis por criar o quadro, desenhando suas bordas e realizando sua impressão.

O quadro foi criado utilizando-se de um função do tipo “*char*”, o que possibilitou trabalhar com os caracteres exigidos.

Tais comentários podem ser observados na imagem a seguir, a **Figura 2**.



```
1 char criaQuadro(char quadro[MAX_LINHAS][MAX_COLUNAS])
2 {
3     for(int i = 0; i < MAX_COLUNAS; i++){
4         quadro[0][i] = '-';
5         quadro[MAX_LINHAS-1][i] = '-';
6     }
7     for(int j = 0; j < MAX_LINHAS; j++){
8         quadro[j][0] = '|';
9         quadro[j][MAX_COLUNAS-1] = '|';
10    }
11    for(int i = 1; i < MAX_LINHAS-1; i++){
12        for(int j = 1; j < MAX_COLUNAS-1; j++){
13            quadro[i][j] = ' ';
14        }
15    }
16
17    return quadro[MAX_LINHAS][MAX_COLUNAS];
18 }
```

**Figura 2 - Screenshot da função citada**

### “simbolos.c”

Neste arquivo estão contidas as principais funções deste trabalho, isto é, as funções de desenhos.

Como havia a necessidade de implementar uma função que foi nomeada de “Todos”, a qual faz todos os três desenhos principais de maneira aleatória, foi feita a opção de fazer a separação da parte que efetivamente realiza os desenhos de cada símbolo, evitando assim, repetições desnecessárias no código.

Para exemplificar tal questão, a seguir está um *screenshot* do arquivo “cabecalhos.h”, na **Figura 3**.

```

1  #define MAX_COLUNAS 80
2  #define MAX_LINHAS 20
3
4  void imprimeQuadro(char quadro[MAX_LINHAS][MAX_COLUNAS]);
5  char criaQuadro(char quadro[MAX_LINHAS][MAX_COLUNAS]);
6  void Menu();
7
8  void desenhaAsterisco(char quadro[MAX_LINHAS][MAX_COLUNAS], int qtd);
9  void Asterisco(int qtd);
10 void desenhaSoma(char quadro[MAX_LINHAS][MAX_COLUNAS], int qtd);
11 void Soma(int qtd);
12 void desenhaX(char quadro[MAX_LINHAS][MAX_COLUNAS], int qtd);
13 void X(int qtd);
14 void Todos(int qtd);
15 void desenhaQuadro(char quadro[MAX_LINHAS][MAX_COLUNAS], int qtd);
16 void Quatro(int qtd);
17
18 void setPosicao(int *x, int *y);
19

```

**Figura 3 - Arquivo “cabecalhos.h”**

A partir da definição aleatória da posição, através da função setPosicao, a qual recebe valores que correspondem às coordenadas x e y da matriz, é realizada uma verificação se o espaço escolhido realmente está livre e a partir de então é realizado o desenho de cada símbolo.

A escolha dessa posição também poderá ser observada a seguir, com a **Figura 4**.

```

1  void setPosicao(int *x, int *y) {
2      *x = 1 + rand() % (MAX_LINHAS - 2);
3      *y = 1 + rand() % (MAX_COLUNAS - 2);
4  }

```

**Figura 4 - função setPosicao**

A seguir será utilizada de exemplo para a explicação anterior a verificação realizada antes do desenho do sinal de adição, como observável na **Figura 5**.

```

1  setPosicao(&pos[0], &pos[1]);
2
3  while (quadro[pos[0]][pos[1]] != ' ' ||
4      quadro[pos[0]-1][pos[1]] != ' ' ||
5      quadro[pos[0]+1][pos[1]] != ' ' ||
6      quadro[pos[0]][pos[1]-1] != ' ' ||
7      quadro[pos[0]][pos[1]+1] != ' ') {
8      setPosicao(&pos[0], &pos[1]);
9  }

```

**Figura 5 - verificação do espaço antes do desenho**

Como resultados foram obtidos o que era esperado como solução para este trabalho. Este apresenta um Menu de fácil compreensão que direciona o usuário ao quadro desejado, permitindo sua visualização e releitura com as mesmas configurações iniciais. Nas **Figuras 6, 7 e 8**, apresentadas a seguir, é possível observar isto.

**Figura 6 - Apresentação do Menu e escolha das opções**





## Referências

- [1] <https://github.com/MatheusPxt21/PAA-TP0>
- [2] [Chat OpenAI](#)