Universidade Federal de Viçosa

Campus Florestal

CCF 330 - Projeto e Análise de Algoritmos

Trabalho Prático 0

Igor Nascimento - 4257

Sumário

1. Introdução

Este trabalho prático envolve a criação de um gerador de arte, onde as imagens serão exibidas em uma grade de 20 linhas por 80 colunas, delimitada por '|' nas colunas e '-' nas linhas. O programa também oferece um menu com cinco opções, permitindo ao usuário escolher entre três tipos de arte, uma mistura aleatória delas, ou uma arte personalizada. O desenvolvimento foi feito no VSCode, com o GitHub usado para versionamento.

2. Desenvolvimento

O desenvolvimento do código se deu através da linguagem C, como solicitado na especificação. O código é dividido entre três arquivos “.c” e dois arquivos “.h”, sendo eles “main.c”, “menu.c”, “quadro.c”, “menu.h” e “quadro.h”. As funções para o quadro e implementação do algoritmo para realizar as obras de arte estão no arquivo “quadro.c”, enquanto o arquivo para apresentar a interface para o usuário estão no “menu.c”. Abaixo segue a figura referente a estrutura do trabalho:

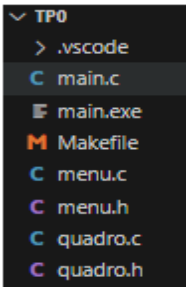

Figura 1: Organização dos arquivos.

2.1 Quadro

Uma matriz foi utilizada para exibir as obras de arte. Dois laços de repetição foram usados para preencher as bordas da matriz com “|” e as linhas laterais de cima e de baixo com “-”. As funções “inicioQuadro”, “criaQuadro” e “printQuadro” foram implementadas, sendo a primeira com a função de realizar uma alocação dinâmica para o início do quadro, a segunda para realizar a criação do quadro e a terceira para apresentar na tela o quadro para o usuário.

2.2 Asterisco/ Soma/ LetraX

Para realizar a criação das artes com Asterisco, Soma e a letra X foi utilizado o mesmo tipo de lógica, basicamente. A função rand( ) foi usada para aleatorizar as posições das figuras. Ademais, na matriz também foi implementado a figura específica para cada opção selecionada, onde uma iteração até a quantidade de figuras escolhidas é gerada. Abaixo segue figura da função Asterisco onde isso é realizado.

```
void Asterisco(char **matriz, int quantidade)
{

    int x, y;

    for (int i = 0; i < quantidade; ++i)
    {
        while (1)
        {

            x = 1 + rand() % 18;
            y = 1 + rand() % 78;

            if (matriz[x][y] == ' ')
            {
                break;
            }
        }
        matriz[x][y] = '*';
    }
}
```

Figura 2: Função que gera Asterisco.

Nesse sentido, para complementar o trabalho prático foi criado a função “verificaEntrada” no arquivo “quadro.c”, onde sua finalidade é verificar a entrada dada pelo usuário e realizar as exceções especificadas. Essa função verifica se o número é maior que 100, caso seja, retorna 100. E, caso seja menor ou igual a zero, a função retorna uma quantidade de figuras aleatórias.

2.3 Obra de arte

Para a obra de arte foram escolhidos emojis feitos com caracteres. Ao implementar os emojis foi necessário armazenar cada caractere em cada espaço da matriz. Foram criados 4 opções de emojis para a obra de arte:

Emoji 1: /(^o^)/
Emoji 2: o_o
Emoji 3:~(-.-)~
Emoji 4: (-_-)

O usuário deve digitar a opção 5 no menu principal para criá-los. A cada vez que o usuário seleciona essa opção é apresentado emojis de quantidade e opções aleatórias, dentro das 4 opções disponibilizadas no código. Para cada emoji foi criado duas funções, uma para inserir e implementar cada parte dele nas posições da matriz e a segunda função foi feita com intuito de realizar a posição aleatória.

O emoji 1 ficou com a função "alegre", o emoji 2 com a função "tedio", emoji 3 sendo a função "serio" e o emoji 4 com a função "susto", onde cada nome de função referencia o humor que o emoji representa. Abaixo seguem figuras que mostram as funções apresentadas juntamente de um *screenshot* da saída do programa após o usuário digitar a opção 5.

```
void alegre(char **quadro, int quantidade)
{
    int x, y;
    srand(time(0));

    for (int i = 0; i < quantidade; ++i)
    {
        while (1)
        {
            x = 1 + rand() % 17;
            y = 1 + rand() % 77;

            if (quadro[x][y] == ' ' && quadro[x][y + 1] == ' ' && quadro[x][y + 2] ==
            {
                break;
            }
        }
        inseriralegre(quadro, x, y);
    }
}
```

Figura 3: Função Alegre.

```
void inseriralegre(char **quadro, int x, int y)
{


    quadro[x][y] = '/';
    quadro[x][y + 1] = '(';
    quadro[x][y + 2] = '^';
    quadro[x][y + 3] = 'o';
    quadro[x][y + 4] = '^';
    quadro[x][y + 5] = ')';
    quadro[x][y + 6] = '/';
};
```

Figura 4: Função inserirAlegre.

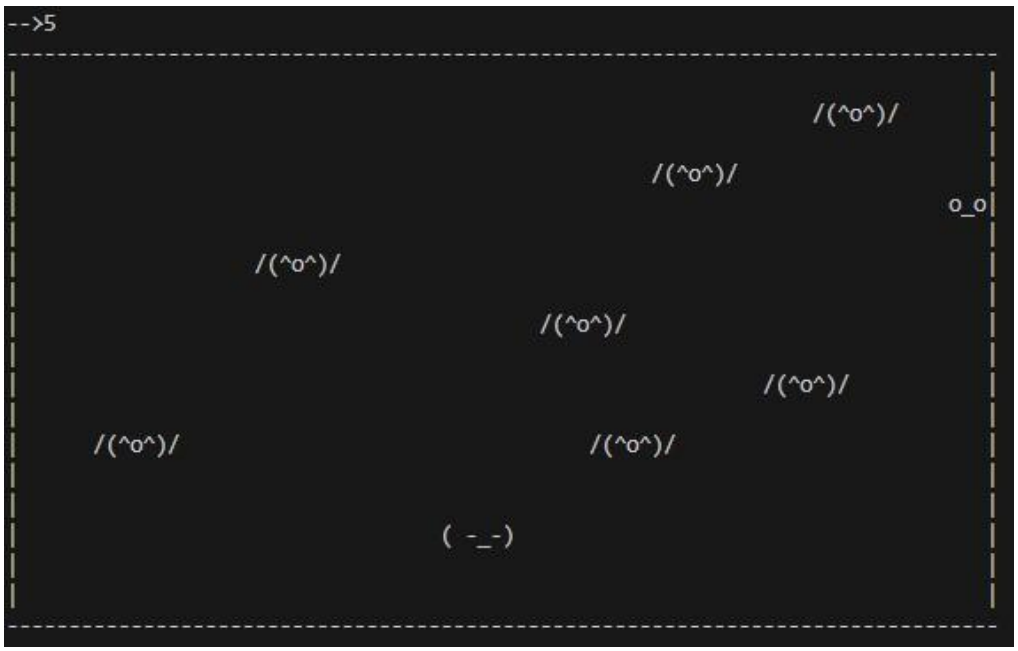

Figura 5: *Screenshot* da opção 5.

3. Execução

Como especificado, foi criado um arquivo makefile para a automação da compilação do arquivo. Segue abaixo figura com o código referente a esse arquivo.

Makefile

```
1  all:
2      gcc main.c quadro.c menu.c -o main
3  run:
4      ./main
5  clean:
6      rm main
```

Figura 6: Arquivo Makefile.

Para compilar o programa, basta digitar no terminal “make” e em seguida digitar “./main”. Após compilar o código, é exibido o menu principal para o usuário. Nesse menu principal contém 5 opções, sendo a primeira referente a obra de arte somente com asteriscos simples, a segunda uma junção de símbolo de soma com asteriscos, a terceira opção letra X com asteriscos, a quarta alternativa é para uma obra de arte com figuras aleatórias e, por fim, a opção cinco, a obra de arte com emojis. Abaixo seguem imagens das opções selecionadas pelo usuário.

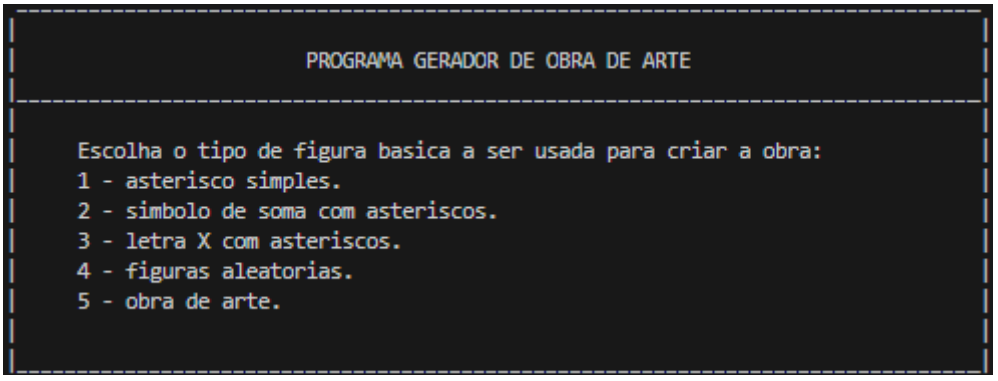

Figura 7: Menu principal.

Figura 8: Opção 1, asteriscos simples.

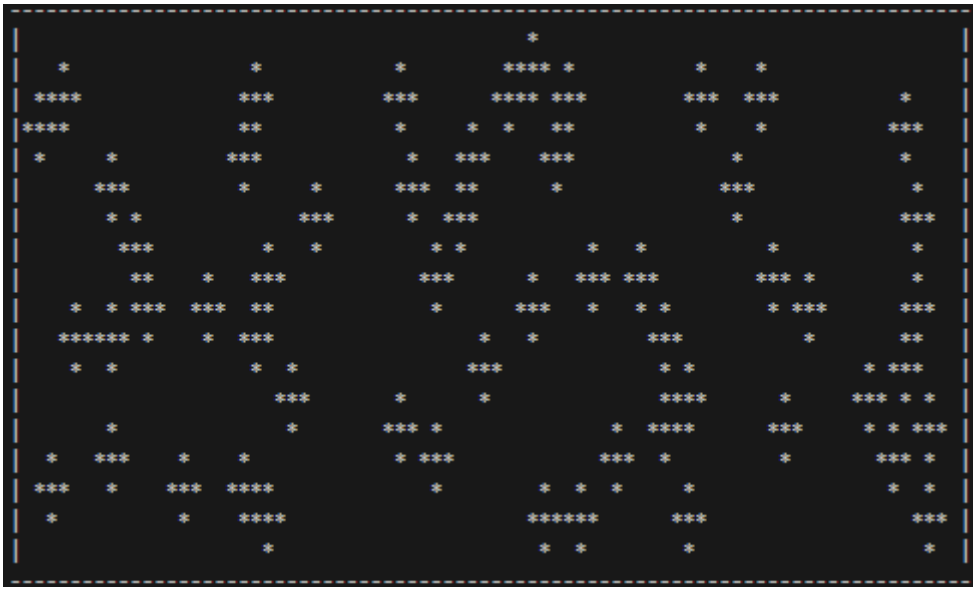

Figura 9: Opção 2, símbolo com soma de asteriscos.

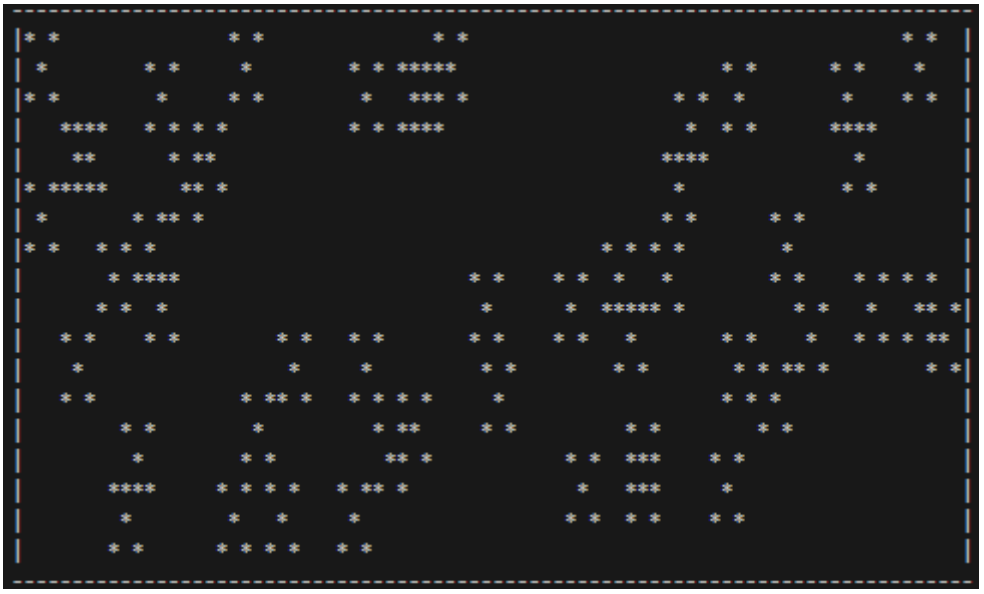

Figura 10: Opção 3, letra X com asteriscos.

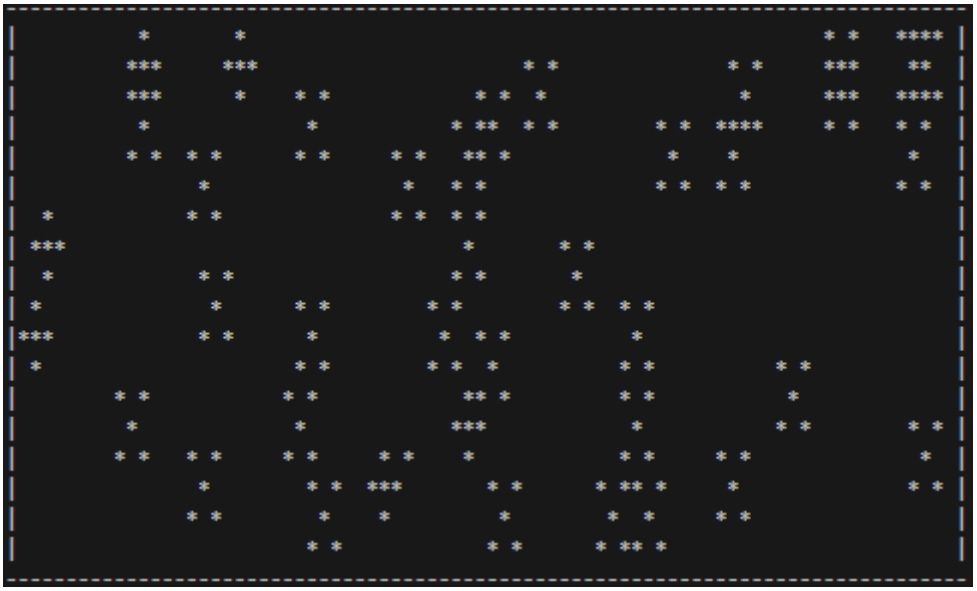

Figura 11: Opção 3, figuras aleatórias.

4. Conclusão

O objetivo principal do trabalho foi alcançado com sucesso: a criação de um programa capaz de gerar obras de arte aleatórias. Durante o desenvolvimento, surgiram dificuldades na inclusão de determinados caracteres para a construção dos emoticons, devido às limitações da linguagem C. No entanto, foram encontradas alternativas que possibilitaram o ajuste do código para atender a essa funcionalidade.Ao longo do processo, houve uma progressão no entendimento da linguagem, o que facilitou a resolução dos desafios, a correção de erros e a otimização dos algoritmos. Assim, todas as especificações exigidas pelo professor foram cumpridas, e o projeto foi entregue conforme planejado. Em conclusão, o programa foi implementado e testado com êxito, confirmando o sucesso da proposta inicial de gerar obras de arte aleatórias.

5. Referências

Ziviani, N. **Projetos de Algoritmos em C**. 3ª edição, São Paulo: Thomson Learning, 2004.