Trabalho Prático 0 - Gerador de Obras de Arte

Vitor M. Andrade

[1]Universidade Federal de Viçosa – Campus Florestal
Departamento de Ciência da Computação
Florestal, MG – Brazil

vitor.mathias@ufv.br

Abstract. *This report presents the development and implementation of a C-language artwork generator. The program creates visual compositions through the random positioning of different geometric figures in a frame delimited by borders. The analysis focuses on the modularization of the code, random number generation, and the inclusion of a creative figure proposed by the student.*

Resumo. *Este relatório apresenta o desenvolvimento e implementação de um gerador de obras de arte em linguagem C. O programa cria composições visuais através do posicionamento aleatório de diferentes figuras geométricas em um quadro delimitado por bordas. A análise foca na modularização do código, uso de números aleatórios e na inclusão de uma figura criativa proposta pelo aluno.*

1. Introdução

O objetivo deste trabalho é desenvolver um programa que gere obras de arte de forma automatizada, combinando conceitos de programação em C, manipulação de matrizes, geração de números aleatórios e modularização através de TADs. O programa imprime figuras dentro de um quadro de 20 linhas por 80 colunas, delimitado por bordas, e oferece ao usuário a possibilidade de escolher entre diferentes tipos de desenhos. Além das figuras obrigatórias (asterisco, símbolo de soma e letra X), foi implementada uma figura opcional, criada pelo aluno, atendendo ao requisito de criatividade do trabalho.

2. Organização

2.1. Arquitetura do Sistema

O programa foi desenvolvido seguindo princípios de modularização, dividido em três arquivos principais:

- **funcoes.h** – declarações, protótipos e constantes.
- **funcoes.c** – implementações das funções de inicialização, borda, impressão e figuras.
- **obras.c** – programa principal, contendo o menu e controle do fluxo.

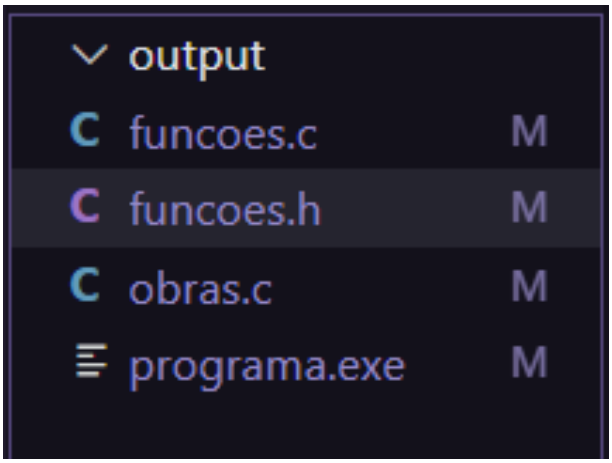

Figura 1. Divisão de arquivos utilizada no projeto

3. Desenvolvimento

3.1. Menu e Fluxo de Execução

O programa inicia exibindo um menu onde o usuário escolhe o tipo de figura e a quantidade desejada. Se o número digitado for menor ou igual a zero, a quantidade de figuras é gerada aleatoriamente entre 1 e 100. Se for maior que 100, o limite é fixado em 100.

```
PROGRAMA GERADOR DE OBRA DE ARTE:
================================
Escolha o tipo de figura basica a ser usada para criar a obra:
1 - asterisco simples
2 - simbolo de soma com asteriscos
3 - letra X com asteriscos
4 - figuras aleatorias
5 ou qualquer outro numero - Letra J
Digite o tipo de figura basica desejada: 4
Digite a quantidade de figuras  (<= 0 para aleatorio): 5
```

Figura 2. Menu de interação do usuário

3.2. Figuras Implementadas

Foram implementadas as seguintes figuras:

- Asterisco simples.
- Símbolo de soma com asteriscos.
- Letra X.
- Figuras aleatórias (mistura das três anteriores).
- Figura proposta pelo aluno – **Letra J**.

Para a escolha da figura 5 era uma prioridade minha fazer algum desenho que envolvesse a religião, a priori, imaginei a estrela de Davi, porém não seria possível fazer ela dentro de uma escala 3x3 (Escala para caber 100 imagens). Portanto, optei pela letra para representar a inicial de Jesus.

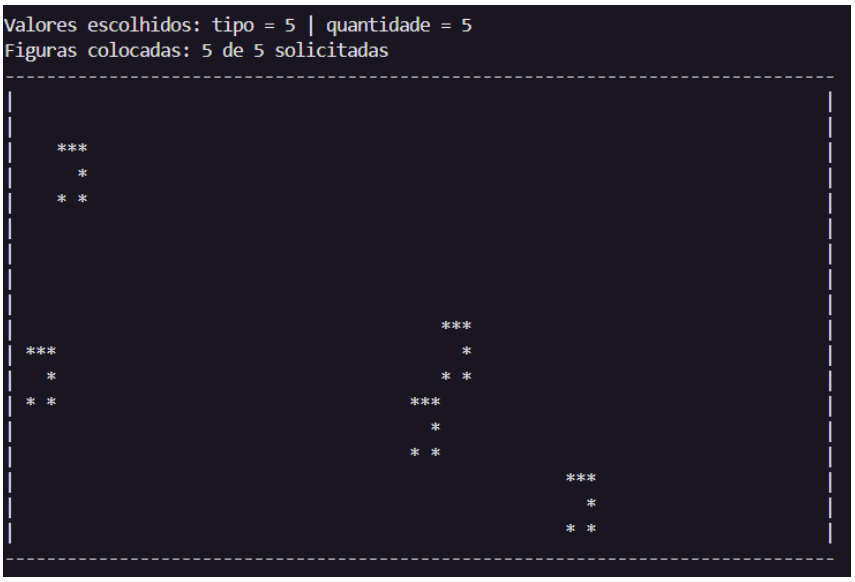

Figura 3. Letra J escolhida para o desenho

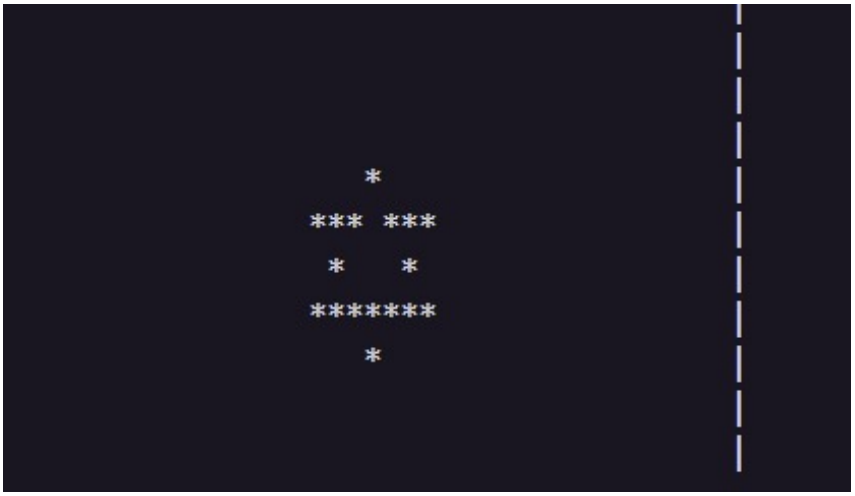

Figura 4. Estrela de Davi anteriormente testada

3.3. Tratamento de Colisão

Para não permitir colisões e um desenho sobrepor o outro foi criado a função VerificaArea, responsável por verificar se as posições que serão ocupadas por uma nova figura estão ocupadas ou não. Caso seja possível (área livre) ela retorna 1, caso contrário retorna 0.

```c
int VerificaArea(int linha, int col, int tipo)
{
    switch (tipo)
    {
    case 1:
        if (linha >= 1 && linha < LINHAS - 1 && col >= 1 && col < COLUNAS - 1)
            return (quadro[linha][col] == ' ') ? 1 : 0;
        break;

    case 2:
        if (linha >= 1 && linha < LINHAS - 3 && col >= 1 && col < COLUNAS - 3)
        {
            if (quadro[linha][col + 1] == ' ' &&
                quadro[linha + 1][col] == ' ' &&
                quadro[linha + 1][col + 1] == ' ' &&
                quadro[linha + 1][col + 2] == ' ' &&
                quadro[linha + 2][col + 1] == ' ')
                return 1;
        }
        break;

    case 3:
        if (linha >= 1 && linha < LINHAS - 3 && col >= 1 && col < COLUNAS - 3)
        {
            if (quadro[linha][col] == ' ' &&
                quadro[linha][col + 2] == ' ' &&
                quadro[linha + 1][col + 1] == ' ' &&
                quadro[linha + 2][col] == ' ' &&
                quadro[linha + 2][col + 2] == ' ')
                return 1;
        }
        break;

    case 5:
        if (linha >= 1 && linha < LINHAS - 3 && col >= 1 && col < COLUNAS - 3)
        {
```

Figura 5. Função VerificaArea

3.4. Repetição de quadros

O programa implementa um loop do-while que permite ao usuário gerar múltiplos quadros com os mesmos parâmetros. Após cada execução, o sistema pergunta se deseja criar um novo quadro mantendo tipo de figura e quantidade. A cada repetição, o quadro é completamente reinicializado e novas posições aleatórias são geradas, permitindo explorar diferentes composições visuais sem necessidade de reinserir os parâmetros iniciais. O loop é encerrado quando o usuário digita qualquer caractere diferente de 's' ou 'S'.

```c
        }

        printf("Figuras colocadas: %d de %d solicitadas\n", figuras_colocadas, qtd);

        imprimir_quadro();

        printf("\nDeseja gerar um novo quadro com os mesmos parametros? (s/n): ");
        scanf(" %c", &refazer);

    } while (refazer == 's' || refazer == 'S');

    printf("\nObrigado por usar o gerador de obras de arte!\n");
    return 0;
}
```

Figura 6. Parte da implementação feita para o looping

4. Resultados e Exemplos

A seguir, são apresentados exemplos de execução do programa no terminal. Cada imagem representa uma saída do programa com diferentes opções escolhidas pelo usuário.

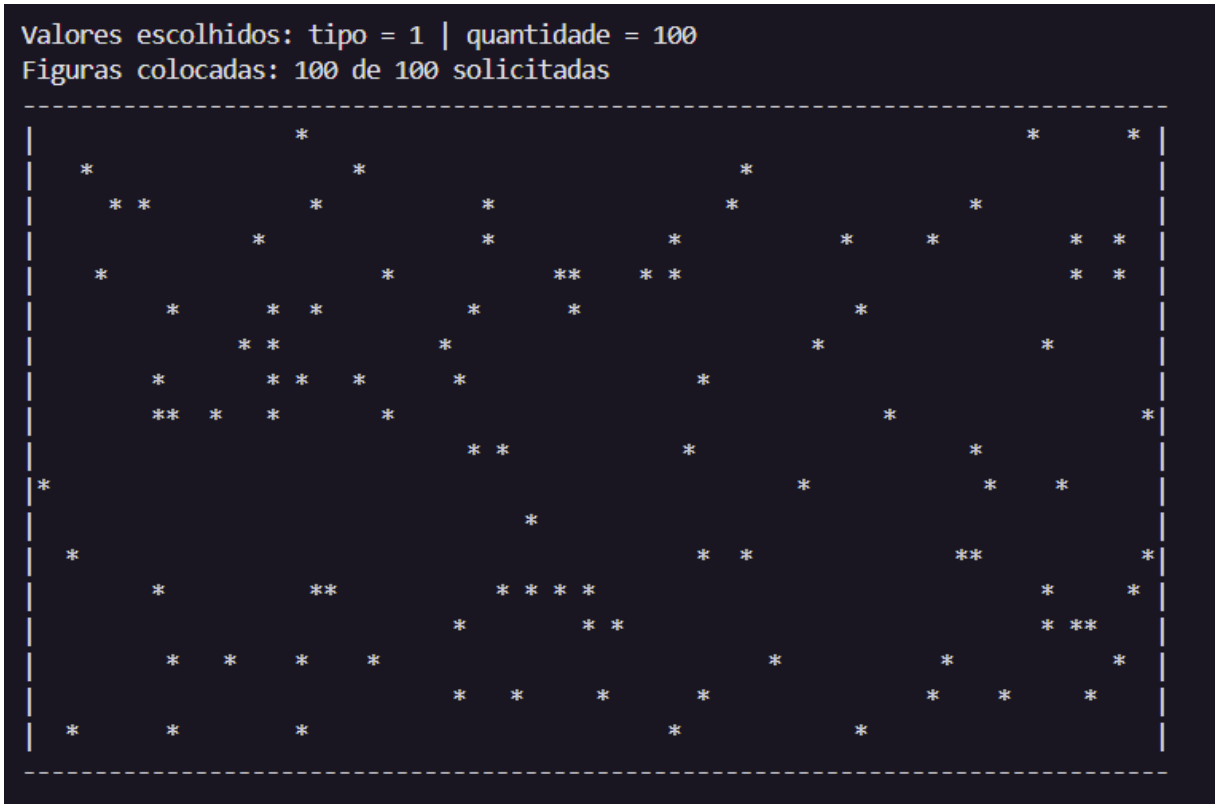

Figura 7. Exemplo de saída com asteriscos simples.

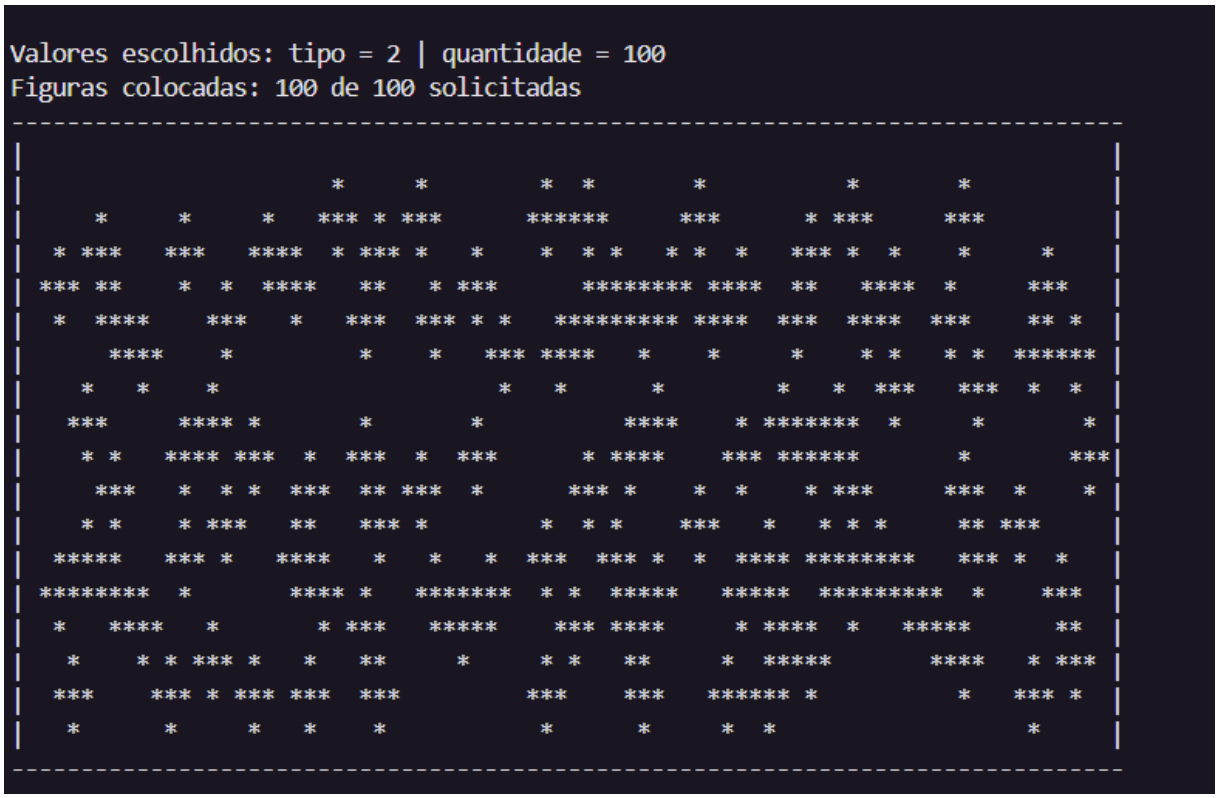

Figura 8. Exemplo de saída com símbolos de soma.

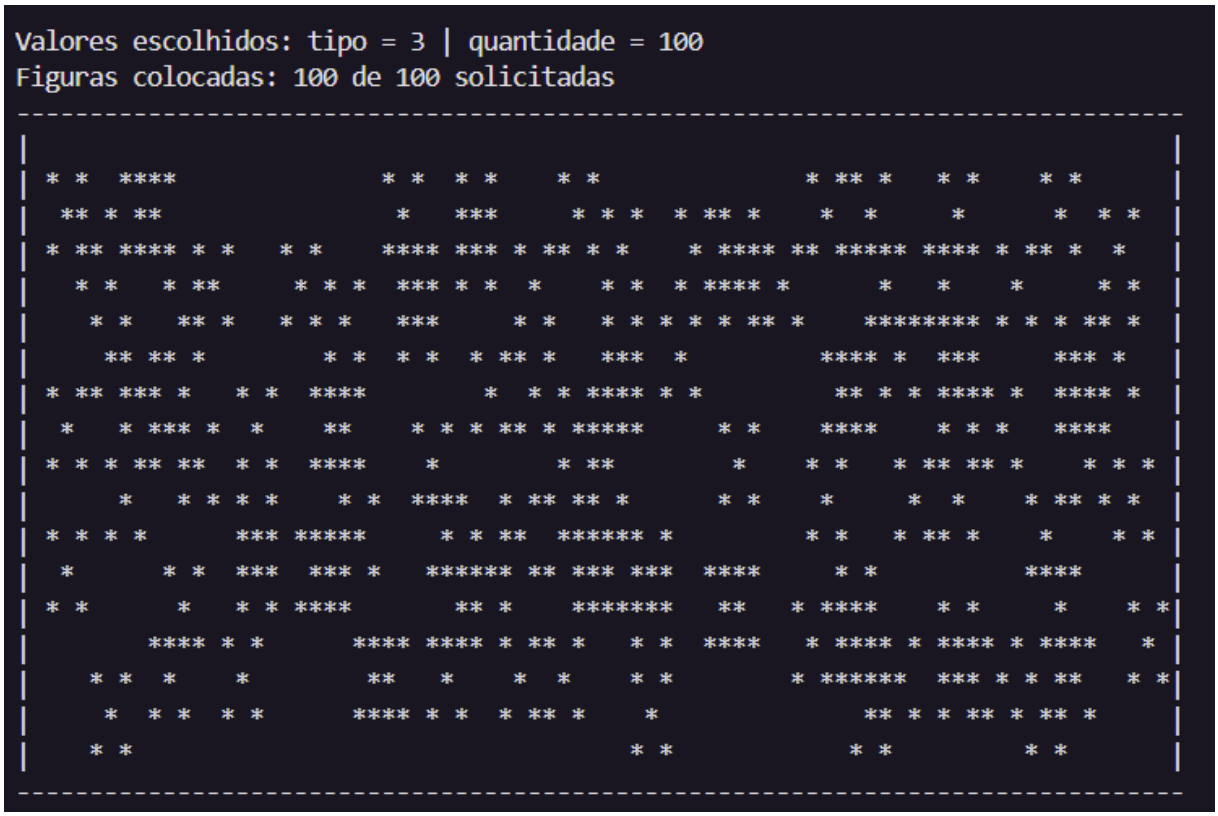

Figura 9. Exemplo de saída com a letra X.

Figura 10. Exemplo de saída com a figuras aleatorias.

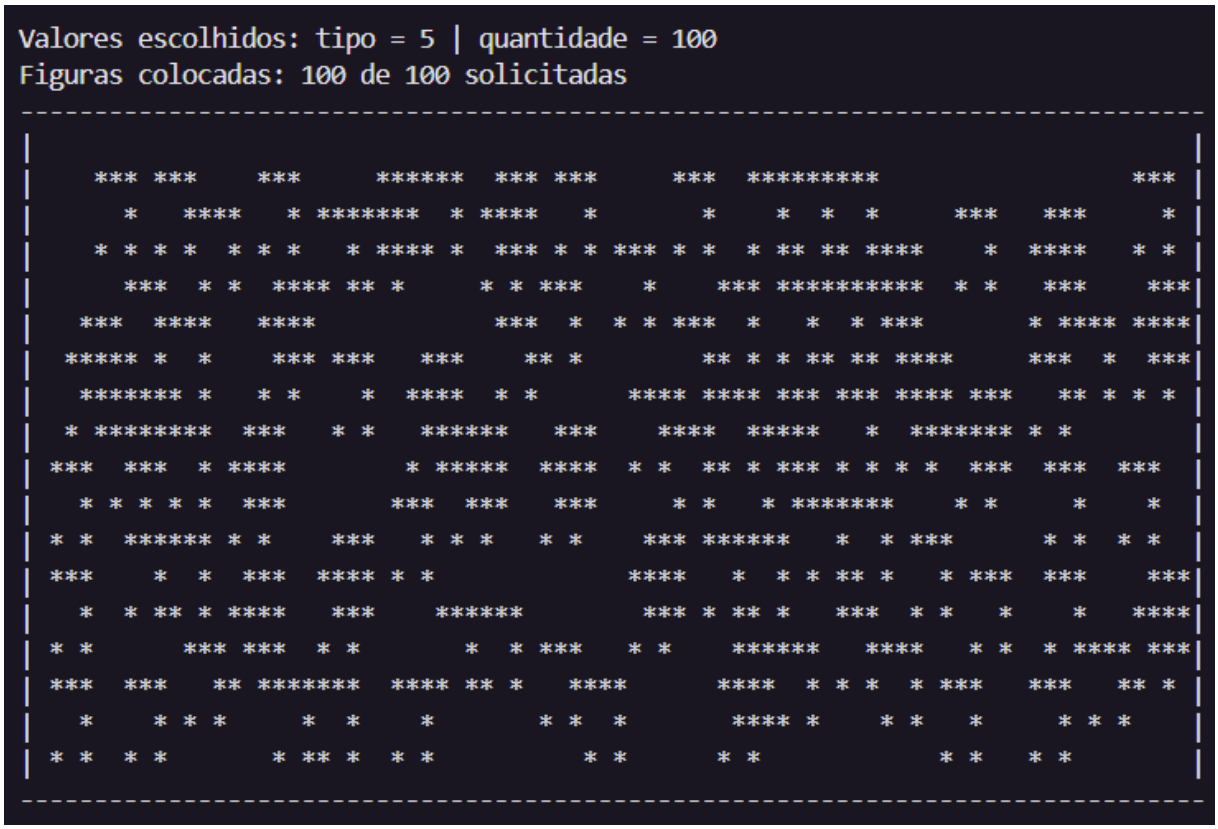

Figura 11. Exemplo de saída com a figura personalizada (Letra J).

5. Conclusão

O trabalho possibilitou a prática de conceitos básicos da linguagem C, como manipulação de matrizes, funções, modularização e uso de números aleatórios. A criação da figura personalizada demonstrou criatividade e consolidou o entendimento sobre a manipulação de índices dentro da matriz.

GitHub

O projeto completo pode ser acessado em: https://github.com/VMathiasAndrade/TP0---PAA.git