



SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE  
FEDERAL DE VIÇOSA · UFV CAMPUS FLORESTAL

## **Trabalho 0 - PAA**

### **Gerador de obras de artes aleatórias**

Miguel Antônio Ribeiro e Silva [EF04680]

Florestal - MG

2022

## **Sumário**

### **1 - Introdução**

1.1 - Como executar

1.2 - Organização

### **2 – Desenvolvimento**

2.1 - Criar\_Matriz()

2.2 - menu()

2.3 - flush\_in()

2.4 - Asteriscos\_Simples()

2.5 - Soma\_Asterisco()

2.6 - X\_Asterisco()

2.7 - Figuras\_Aleatorias()

2.8 - Arte\_Colorida()

2.9 - Printar\_Matriz()

2.10 - main()

### **3– Referências**

### **4- Conclusões**

## 1. Introdução

O projeto consiste em gerar uma obra de arte aleatória implementado em linguagem C usando, para isso, funções que randomizam números, a fim de imprimir no terminal do usuário um quadro 20x80 artístico e personalizável. O programa utiliza asteriscos e o caractere full block (█) para a geração de imagens, auxiliadas por uma matriz de inteiros, alocada estaticamente.

### 1.1. Como executar

1. Pré-requisitos: Linux ou Windows com WSL, GCC, Make (opcional).
2. Abra o terminal do Linux ou WSL e digite make para compilar o projeto e make run para executar. Caso não possua o comando make, use **gcc -o main funcoes.c main.c** e **./main** para executar.
3. Siga o menu do programa.

### 1.2 Organização

Na Figura 1 é possível visualizar a organização do projeto. O arquivo principal **main.c** contém as bibliotecas utilizadas no projeto, declaração das variáveis necessárias e o chamado para o menu e as funções necessárias para a geração da imagem, codificadas nos arquivos **funcoes.c** e **funcoes.h**; **makefile** com os comandos necessários para compilar os códigos; **readme.txt** detalha os comandos necessários para executar o código da maneira correta. Para versionar o projeto, foi utilizado o github[1].

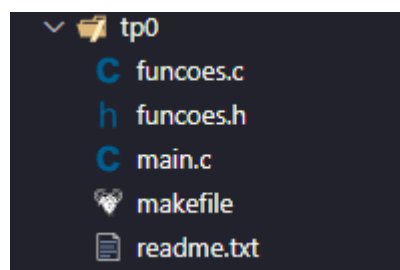


Figura 1 - repositório do projeto

## 2. Desenvolvimento

Primeiramente, o programa aloca uma matriz de inteiros 17x77 e a inicializa com 0 em todas as posições , após isso um menu interativo é chamado e o usuário escolhe que tipo de imagem deseja gerar. A partir dessa escolha, o código executa diversas funções, modificando a matriz e inserindo números inteiros aleatórios, função rand() e srand() [3] que auxiliarão na geração da arte. No fim, o programa imprime a imagem e o usuário escolhe se deseja encerrar ou continuar e fazer modificações. Veja a seguir, o que cada função é responsável por fazer e como foram criadas.

### 2.1. Criar\_Matriz()

Aloca uma matriz de inteiros 17x77 e a inicializa com 0 em todas as posições.



### 2.2 menu()

Cérebro do programa, é aqui que o usuário interage e escolhe como deseja gerar sua obra de arte (**Figura 2**).

```
PROGRAMA GERADOR DE OBRA DE ARTE
=====
1 - Asterisco simples
2 - Simbolo de soma com asteriscos
3 - Letra X com asteriscos
4 - Figuras aleatorias
Outro - Arte Colorida
=====
Digite o numero da opcao desejada:
```

Figura 2 – o menu

O menu foi elaborado da seguinte maneira:

- O usuário deverá escolher uma das opções e o número de imagens que deseja em sua arte
- Opção 1 - asterisco simples: gera figuras com símbolo “\*” através da função `Aterisco_Simples()`.
- Opção 2 - símbolo de soma com asteriscos: gera figuras com o símbolo  através da função `Soma_Asterisco()`.
- Opção 3 – letra X com asteriscos: análogo a anterior  e utiliza a função `X_Asterisco()`.
- Opção 4 – mistura aleatoriamente as três figuras e a figura colorida através da função `Figuras_Aleatorias()`
- Outro – gera uma arte colorida, `Arte_Colorida()`.

Após a geração da figura, o programa imprime na tela o quadro e pergunta se o usuário deseja encerrar ou modificar (Figura 3).

```
Digite uma das opcoes abaixo:
1 - Refazer o quadro com mesmos valores |exclui as figuras anteriores!|
2 - Refazer com novos valores
3 - Inserir novas figuras no quadro
Outro - Sair
```

**Figura 3 – o menu**

- Opção 1 – o algoritmo aproveita os valores escolhidos anteriormente pelo usuário e cria uma nova figura com os mesmos valores.
- Opção 2 – cria uma nova figura, com valores diferentes.
- Opção 3 – insere novas figuras no quadro já gerado anteriormente
- Opção 4 – encerra o programa e agradece pelo uso.

## 2.3 `flush_in()`

Limpa o buffer do teclado a fim de evitar erros[2].

## 2.4 Asteriscos\_Simples()

Essa função é responsável por gerar imagens com apenas um asterisco simples em cada posição da matriz e funciona da seguinte maneira: seleciona aleatoriamente uma posição na matriz de zeros, função **rand()**, verifica se ela realmente está ocupada por um número 0, se sim, coloca nessa posição o **número 1**, que significa que ali deverá ser impresso um asterisco (**Figura 4**), se não, seleciona novamente uma opção aleatória e assim por diante, para o número de figuras desejadas pelo usuário (**Figura 5**)

```
do{
    int x = rand() % 17;
    int y = rand() % 77;
    if(matriz[x][y] == 0){
        matriz[x][y] = 1;
        break;
    }
}
while(1);
```

Figura 4 - comparações

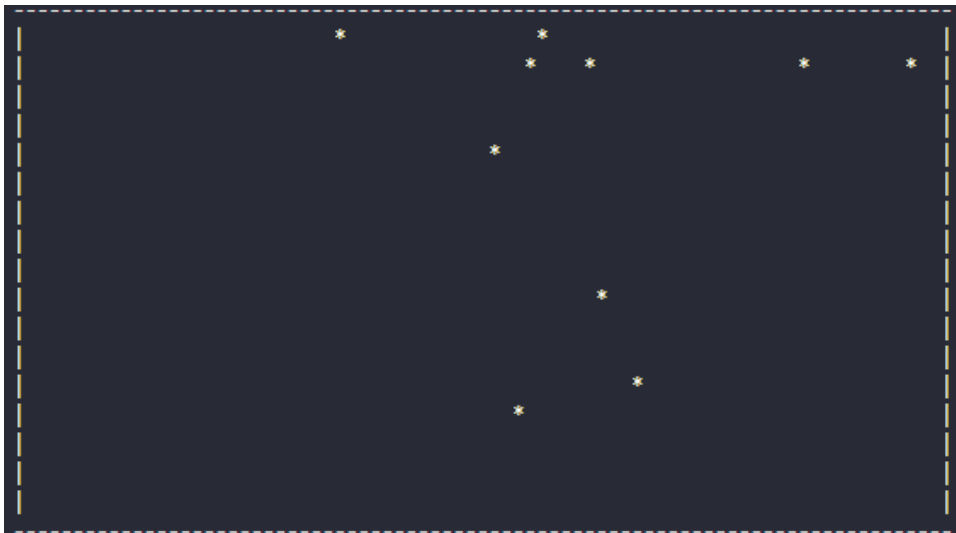


Figura 5 – arte gerada com 10 asteriscos simples

## 2.5. Soma\_Asterisco()

Análoga a anterior (usando a mesma lógica de 0s e 1s), verifica as posições necessárias, geradas aleatoriamente, para a inserção do símbolo + com asteriscos. Veja que as figuras não ficam sobrepostas, apesar de parecer que estão **(Figura 6)**.



Figura 6 – arte gerada com 20 cruzes com asteriscos

## 2.6. X\_Asterisco()

A figura agora, é um X com asteriscos, e as premissas são as mesmas para a criação e modificação na matriz de inteiros **(Figura7)**.

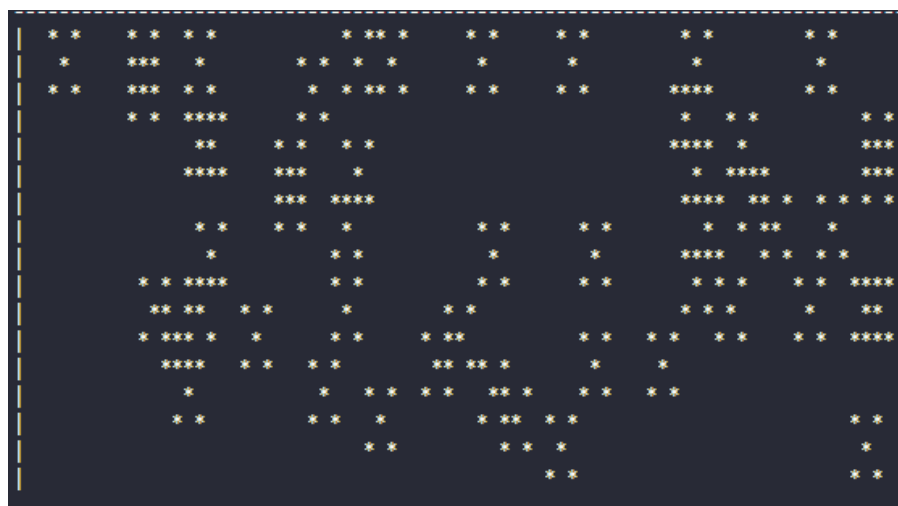


Figura 7 – arte gerada com 50 figuras X-Asterisco.

## 2.7. Figuras\_Aleatorias()

Nessa função, o quadro é totalmente aleatório, até mesmo as figuras que o compoñham. Utiliza as figuras, exemplificadas acima, e também a figura especial, o caractere colorido (seção 2.8), como mostra a **Figura 8**.

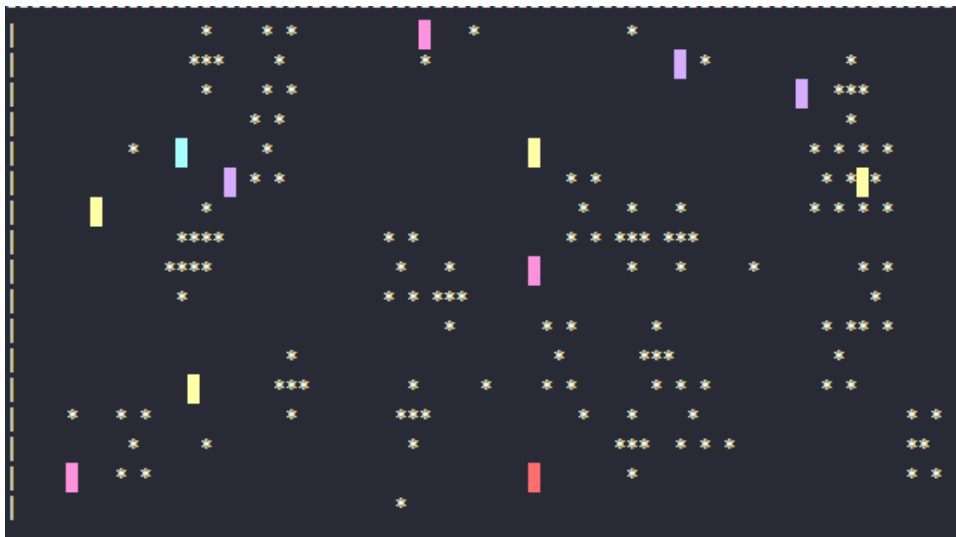


Figura 8 – arte com 50 figuras aleatórias

## 2.8. Arte\_Colorida()

Criada por mim, a arte colorida é composta por um caractere, chamado full block, “█”. O motivo dessa escolha foi porquê o símbolo ocupa um espaço inteiro, ficando muito interessante e bonito quando impresso no terminal. Para dar cor a esse caractere foram usadas algumas funções do C que colorem os textos do terminal [4] (**Figura 9**); as cores possíveis que podem ser sorteadas são, Rosa, Amarelo, Verde, Branco, Azul, Vermelho, Ciano ou nenhuma, sendo assim, um **espaço vazio**.

Ao seleccionar essa opção pelo menu, o usuário escolhe se quer uma arte completa (**Figura 10**), ou apenas a quantidade solicitada(**Figura 11**). Ao seleccionar, arte completa, o programa retornará uma arte abstrata colorida e encerrará. A lógica é parecida com as anteriores, só que no lugar de 1s é inserido na matriz, o número 2, verificando antes se o lugar, gerado randomicamente, é composto por 0.



```

void Red(){
    printf("\033[1;31m");
}

void Green(){
    printf("\033[1;32m");
}

void White(){
    printf("\033[0m");
}

```

Figura 9 – exemplo de funções que dão cor ao terminal

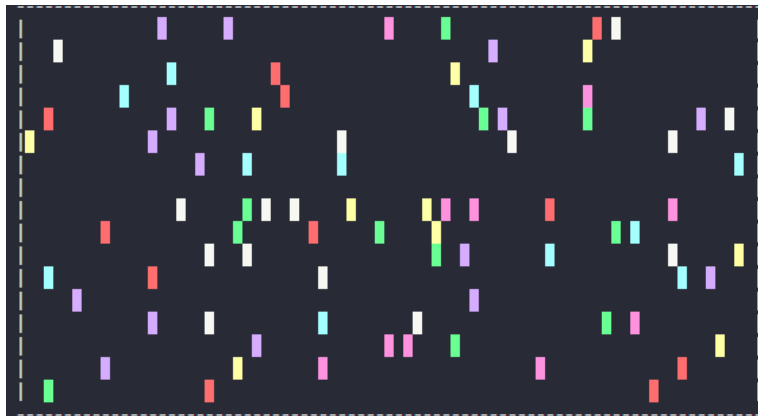


Figura 10 – arte colorida não completa

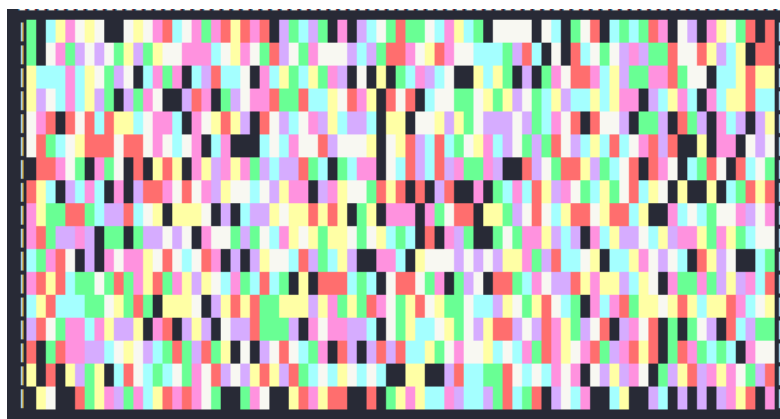


Figura 11 – arte colorida completa

## 2.9. Printar\_Matriz()

Ao fim de cada imagem gerada, o programa imprimirá na tela o quadro 20x80, contendo nele, a matriz 17x77. As bordas de cima e de baixo são compostas pelo símbolo “-”, as laterais “|” e o centro, a matriz de inteiros (0s, 1s, 2s).

No momento da impressão da matriz, o programa verifica se na posição da vez, o lugar é ocupado por 0, 1 ou 2 (**Figura 12**); se for 0, espaço vazio, 1, asterisco e 2, um full block colorido, sendo sua cor, escolhida arbitrariamente, na hora da impressão.

```
if(matriz[i][j] == 0){
    printf(" ");
}
else if(matriz[i][j] == 1){
    printf("*");
}
else if(matriz[i][j] == 2){
    int cor = rand()%8;
    if(cor == 0){
        Red();
        printf("█");
        White();
    }
}
```

Figura 12 – logica da função Printar\_Matriz()

## 2.10. main():

A função **main** é o ponto de partida para a execução do programa. Aqui é chamada a função, **srand(time(NULL))**, Criar\_Matriz(), menu(), Printar\_Matriz(), além dos includes necessários.

### 3. Referências

- [1] [PAA-works/tp0/tp0 at main · miguelribeirokk/PAA-works \(github.com\)](#)
- [2] [c - Limpeza do buffer do teclado após scanf - Stack Overflow em Português](#)
- [3] [C library function - rand\(\) \(tutorialspoint.com\)](#)
- [4] [Como Alterar a Cor do Texto em um Programa C \(wikihow.com\)](#)
- [5] [C Reference function srand\(\) initialize random number generator » CodingUnit Programming Tutorials](#)

### 4. Conclusões

Através desse trabalho foi possível aprender mais sobre as funções rand() e srand(), conhecer e aprender como as cores no terminal são modificadas, limpeza de buffer e manipulação de matrizes para evitar sobreposições. Acredito que com esse trabalho introdutório de aquecimento, a disciplina de Projeto e Análise de Algoritmos fluirá melhor.