



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

Trabalho 0 – Projeto e Análise de Algoritmos

Aquecimento na Linguagem C

Ana Luísa Moreira Rodrigues [5389]

Florestal - MG

2024

Sumário

Sumário

Sumário	2
1. Introdução	3
2. Organização	3
3. Desenvolvimento	4
5. Resultados	8
6.Conclusão	14

1. Introdução

Neste projeto foi proposto a criação de um programa em C capaz de gerar, a partir de certas especificações do usuário, obras de arte aleatórias em um quadro de tamanho de 20 linhas por 80 colunas preestabelecido.

O programa deve conter 4 opções de obras de arte padrões e uma obra de arte específica, inventada pela própria aluna e, a partir das escolhas, posicioná-las aleatoriamente no quadro, seguindo regras que garantam a integridade e a exclusividade das formas. O objetivo é relembrar e retomar conhecimentos sobre lógica de programação e sintaxe da linguagem C já obtidos no decorrer do curso, para que possamos prosseguir com a matéria de Projeto e Análise de Algoritmos.

2. Organização

Para a organização do projeto não houve a necessidade de separar os arquivos de mesmo tipo em pastas diferentes, visto que há somente um arquivo de cada tipo.

Na Figura 1 é possível visualizar a organização do projeto. Na pasta ./TP0_PAA se encontra a pasta de todos os arquivos do código e o arquivo de execução makefile

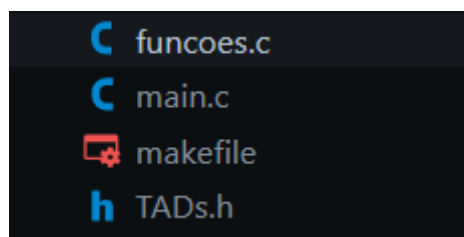


Figura 1 – Organização dos arquivos.

Para executar o projeto, foi utilizado um arquivo Makefile com os comandos necessários para compilar e executar os códigos. Exemplo de execução com Makefile:

Abrir o terminal na pasta do arquivo makefile e digitar: make all (Figura 2).

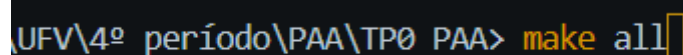
A imagem mostra uma linha de comando em um terminal. O prompt de usuário é 'UFV\4º período\PAA\TP0_PAA>' e o comando digitado é 'make all' seguido de um cursor piscando.

Figura 2 – Makefile.

3. Desenvolvimento

Inicialmente, foram identificados os diferentes níveis de abstração envolvidos no projeto. Isso incluiu a definição de como os números aleatórios seriam gerados, o tratamento da estrutura do quadro para permitir a inserção da quantidade de figuras escolhidas pelo usuário, e a concepção das obras de arte que seriam criadas.

3.1 Geração dos valores aleatórios

A geração de números aleatórios é fundamental no programa, sendo usada para definir as posições das figuras no quadro e algumas características das obras de arte. Em diversas funções, esses números são utilizados para garantir que as figuras sejam posicionadas de forma válida, sem sobreposição. Na função `Escolhe_Arte`, se o número de figuras escolhido pelo usuário for menor ou igual a zero, o programa cria uma quantidade aleatória de figuras entre 1 e 100, conforme solicitado na especificação do trabalho.

Os números pseudoaleatórios são gerados pela função `rand()` da biblioteca `<stdlib.h>`, com a semente inicializada por `srand()` e `time()` da biblioteca `<time.h>` para garantir variação a cada execução. A função `rand()` é usada para definir posições no quadro, determinar o número de figuras e escolher entre diferentes tipos de figuras e cores, com os valores ajustados pela operação de **módulo** (%) para se adequarem aos intervalos corretos.

3.2 Inserção das figuras

Foram criadas structs como `Quadro` e `Cores`, que armazenam informações sobre o tamanho e a matriz do quadro e das cores, respectivamente. Para inserir as figuras, foi utilizada uma operação de “resto de divisão” (%19 ou %79), para darmos limite a posição gerada, e em seguida foi verificado se o espaço no quadro está disponível e, caso esteja, a figura é desenhada, garantindo que não haja sobreposição. operação de “resto de divisão” (%19 ou %79), para darmos limite ao valor gerado.

```

void Gerar_Asterisco(Quadro *quadro, Resposta_Usuario resposta);
void Gerar_Soma(Quadro *quadro, Resposta_Usuario resposta);
void Gerar_X(Quadro *quadro, Resposta_Usuario resposta);
void Gerar_Calopsita(Quadro *quadro, Resposta_Usuario resposta);
void Gerar_Aleatoriamente(Quadro *quadro, Resposta_Usuario resposta);
void Gerar_Calopsita_Cor_(Quadro *quadro, Resposta_Usuario resposta, Cores *cores);

```

Figura 3 – Funções responsáveis por gerar cada tipo de arte.

Com os valores devidamente tratados, é feito a comparação de cada posição da matriz que a figura ocuparia tomando de base o valor gerado como o centro da figura, na prática é simples de ver se utilizarmos coordenadas x e y em uma malha de matriz.

x-1, y-1	x, y-1	x+1,y-1
x-1, y	x,y	x+1, y
x-1, y+1	x, y+1	x+1,y-1

Figura 4 – Malha de matriz.

Conforme mostrado na Figura 4, ao tomar um valor central como referência, é fácil identificar quais posições da matriz precisam ser avaliadas. Essa abordagem facilitou bastante a realização das comparações necessárias para verificar se a inserção da figura era possível ou se seria preciso alterar a posição central.

3.3 Criação de obra de arte específica

Além das figuras básicas solicitadas (um "*", um "X" e um símbolo de soma "+", todos formados por "*"), foi proposto que a aluna criasse obras de arte de forma livre, desde que utilizasse números aleatórios para compô-las. A criação desenvolvida neste projeto foi:

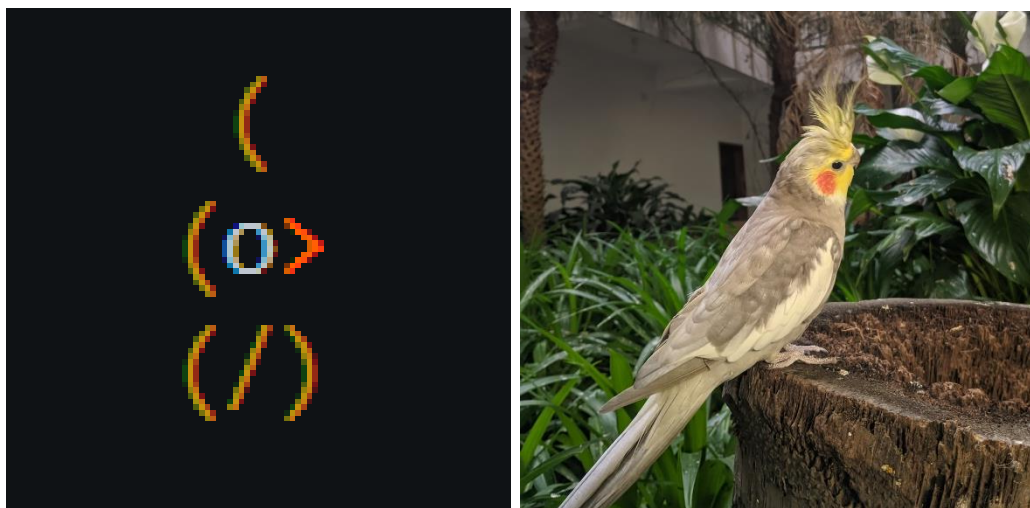


Figura 5 – Arte criada e inspiração.

A obra de arte criada representa uma calopsita, inspirada no meu pássaro preferido, Nino. A figura foi desenhada com dimensões 9x9, de forma proposital, para garantir que a delimitação do quadro de 20x80 não interfira mesmo quando o usuário solicitar o número máximo de figuras (100). Para enriquecer o trabalho e manter a ideia de aleatoriedade, foram desenvolvidas duas opções para a representação da calopsita (opções 5 e >=6).

```
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 - Calopsitas de uma cor (azul, amarelo e verde) aleatoria.
6 ou qualquer outro numero - Calopsitas de uma cor (azul, amarelo e verde) aleatoria cada.
```

Figura 6 – Menu inicial do programa

Na opção 5, é gerada uma quantidade X de calopsitas, todas em uma única cor, que é escolhida aleatoriamente. Já na opção 6, cada calopsita é desenhada com uma cor aleatória diferente, exigindo uma lógica mais elaborada. Para implementar essa variação de cores, foi criada uma segunda matriz, com as mesmas dimensões do quadro, que armazena os valores referentes às cores atribuídas a cada calopsita. O código usa a função `AplicarCor()` para aplicar cores predefinidas (amarelo, azul e verde), variando de acordo com o valor aleatório gerado para cada figura. O uso de structs, como `Quadro` e `Cores`, facilita o gerenciamento tanto da posição das figuras quanto das cores associadas a elas.

Na função `Gerar_Calopsita_Cor_Individual`, apenas as partes correspondentes às penas no desenho são registradas na matriz de cores, enquanto as posições relativas ao bico e ao olho permanecem inalteradas, mantendo suas cores originais. Assim, durante a impressão, as cores são aplicadas com base nas informações contidas nessa matriz.

```
while (posicoes_validas != resposta.Qtd_Figura)

int arte = rand() % 3;

if (quadro->matriz[linha][coluna] == ' ' && quadro->matriz[linha][coluna+1] == ' ' && quadro->matriz[linha][coluna-1] == ' '
    && quadro->matriz[linha+1][coluna-1] == ' ' && quadro->matriz[linha+1][coluna] == ' ' && quadro->matriz[linha+1][coluna+1] == ' '
    && quadro->matriz[linha-1][coluna-1] == ' ' && quadro->matriz[linha-1][coluna+1] == ' ' && quadro->matriz[linha-1][coluna] == ' '
    && coluna >= 0 && linha >= 0)
{
    quadro->matriz[linha][coluna] = 'o';
    quadro->matriz[linha][coluna+1] = '>';
    quadro->matriz[linha][coluna-1] = '(';
    quadro->matriz[linha-1][coluna-1] = ' ';
    quadro->matriz[linha-1][coluna] = '(';
    quadro->matriz[linha+1][coluna] = '/';
    quadro->matriz[linha+1][coluna-1] = '(';
    quadro->matriz[linha+1][coluna+1] = ')';
    quadro->matriz[linha-1][coluna+1] = ' ';

    //Preenche a matriz de cores para identificar onde está cada figura
    cores->matriz[linha][coluna-1] = arte;
    cores->matriz[linha-1][coluna] = arte;
    cores->matriz[linha+1][coluna] = arte;
    cores->matriz[linha+1][coluna-1] = arte;
    cores->matriz[linha+1][coluna+1] = arte;

    posicoes_validas++;
}
if (resposta.Tipo_Arte == 4 && posicoes_validas == 1)
{
    break;
}
```

Figura 7 – Lógica da função que gera a arte da opção 6.

5. Resultados

O código foi desenvolvido em um computador utilizando o sistema operacional Windows 11.

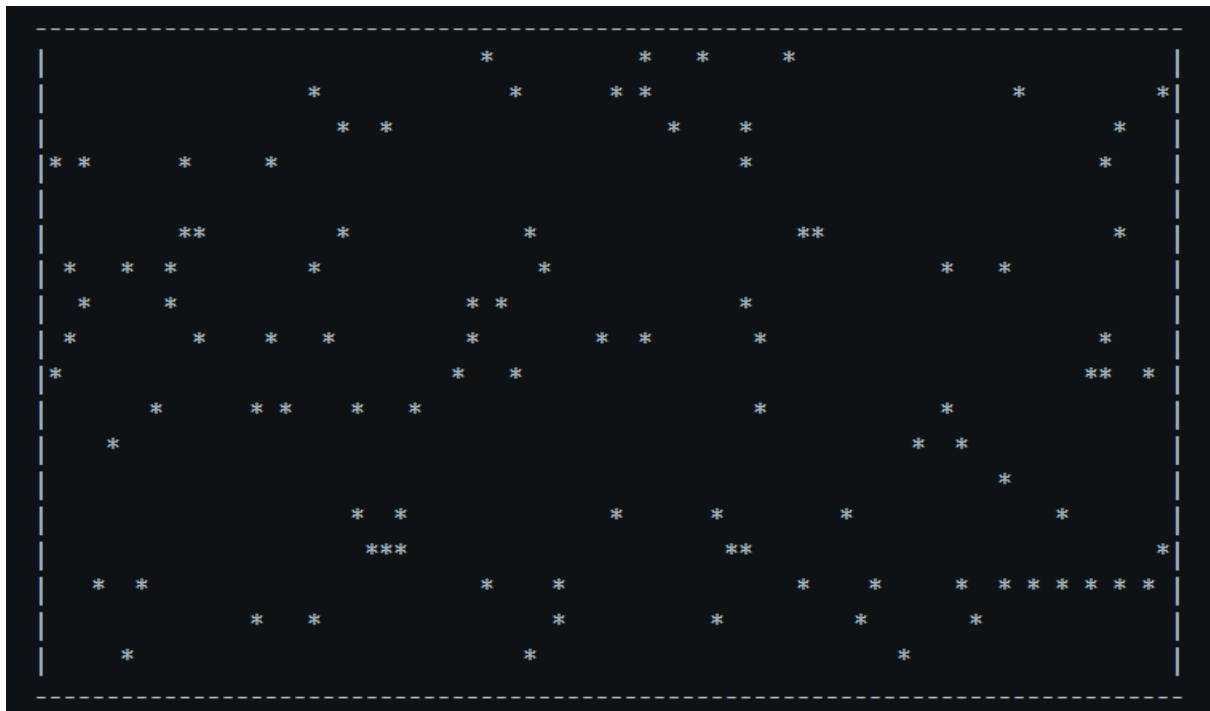


Figura 8.1 – Primeira opção "Asterisco", 100 figuras.



Figura 8.2 – Primeira opção "Asterisco", 100 figuras.

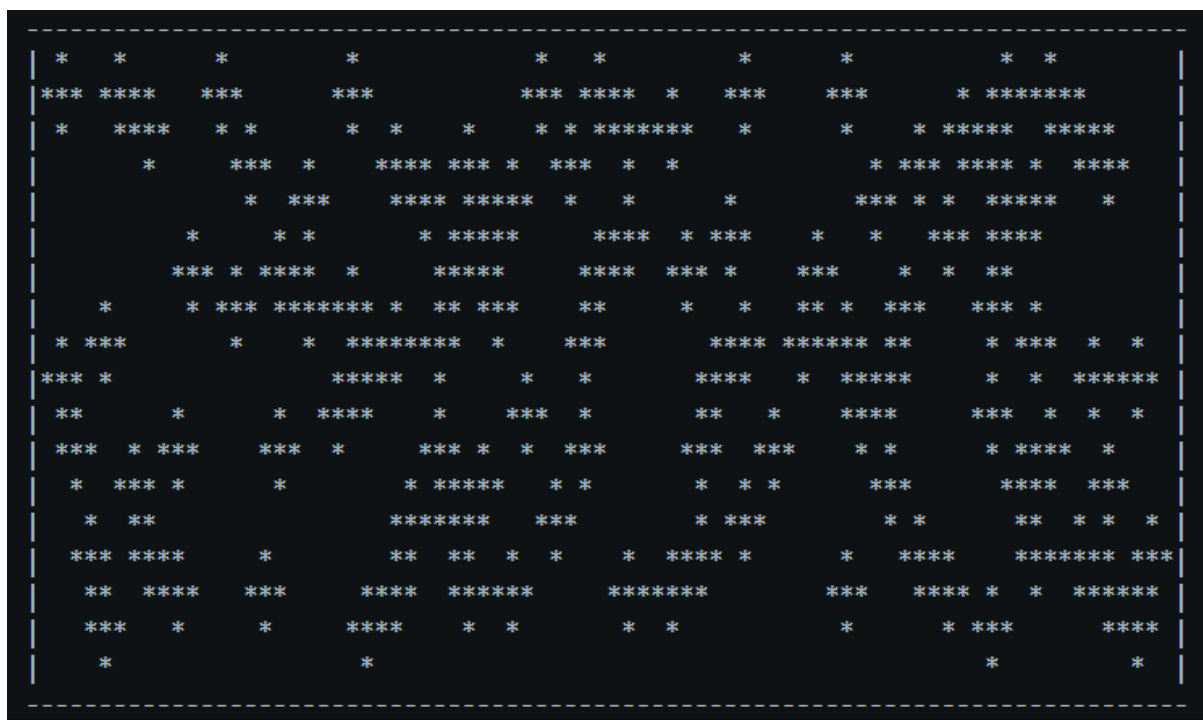


Figura 9.1 – Segunda opção "Soma", 100 figuras.

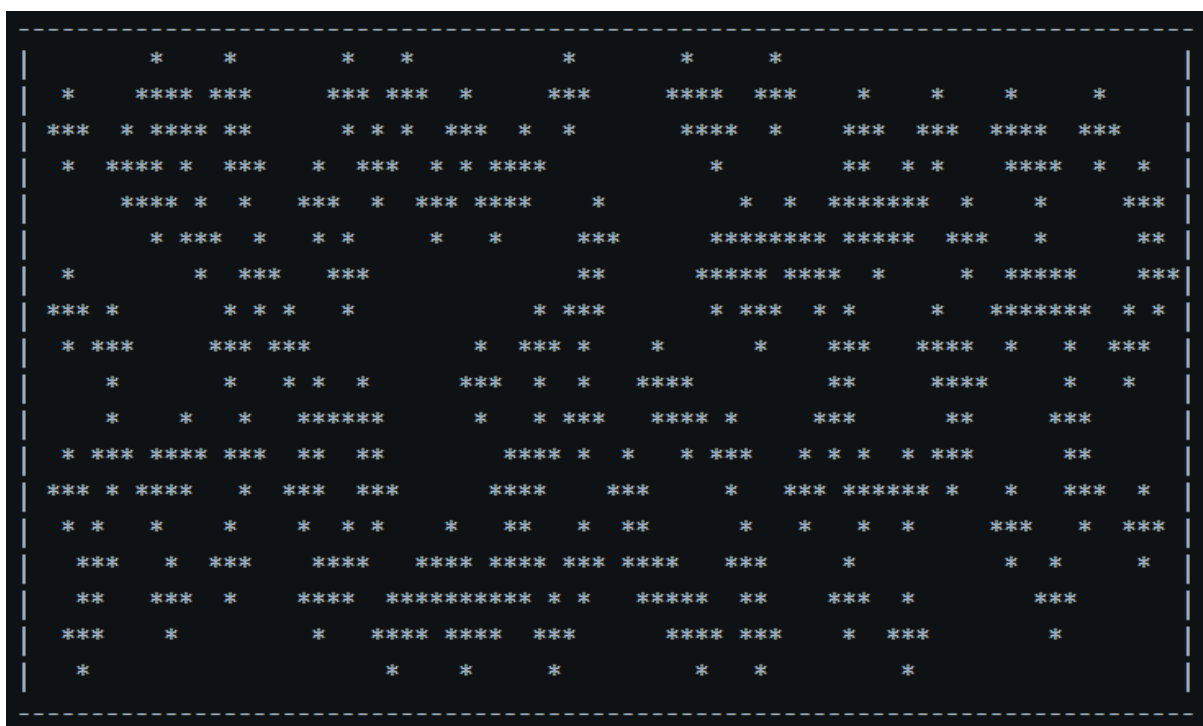


Figura 9.2 – Segunda opção "Soma", 100 figuras.

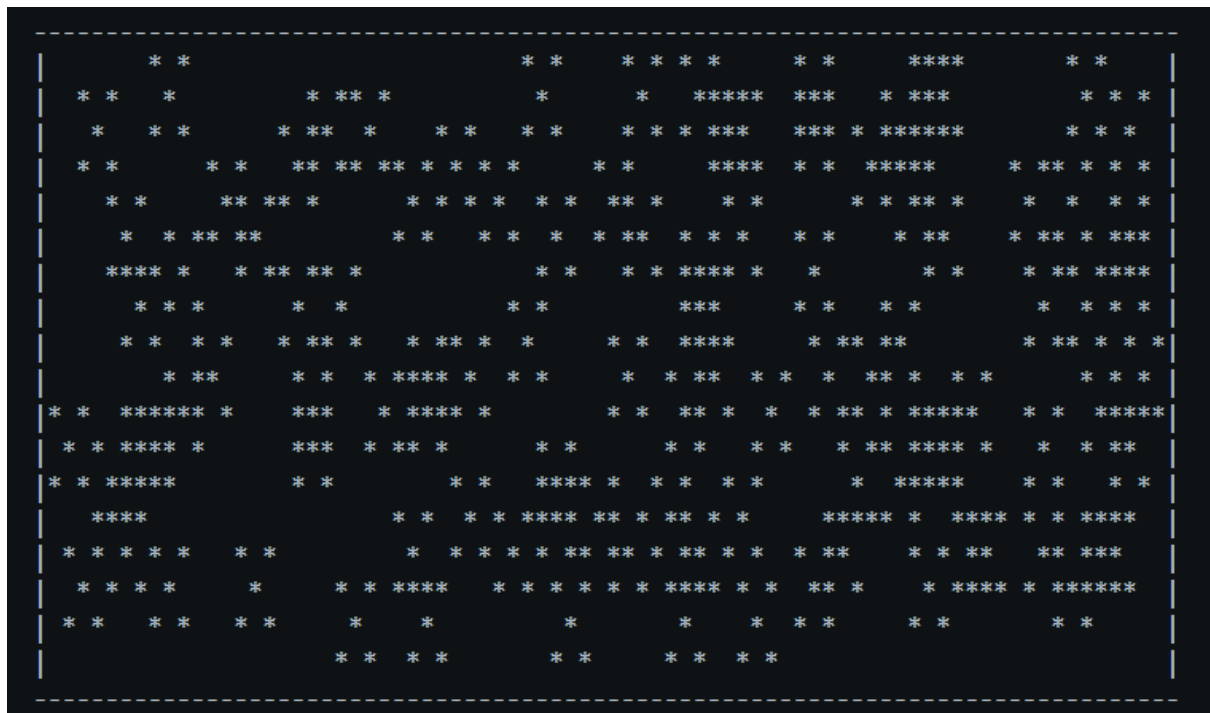


Figura 10.1 – Terceira opção "X", 100 figuras.

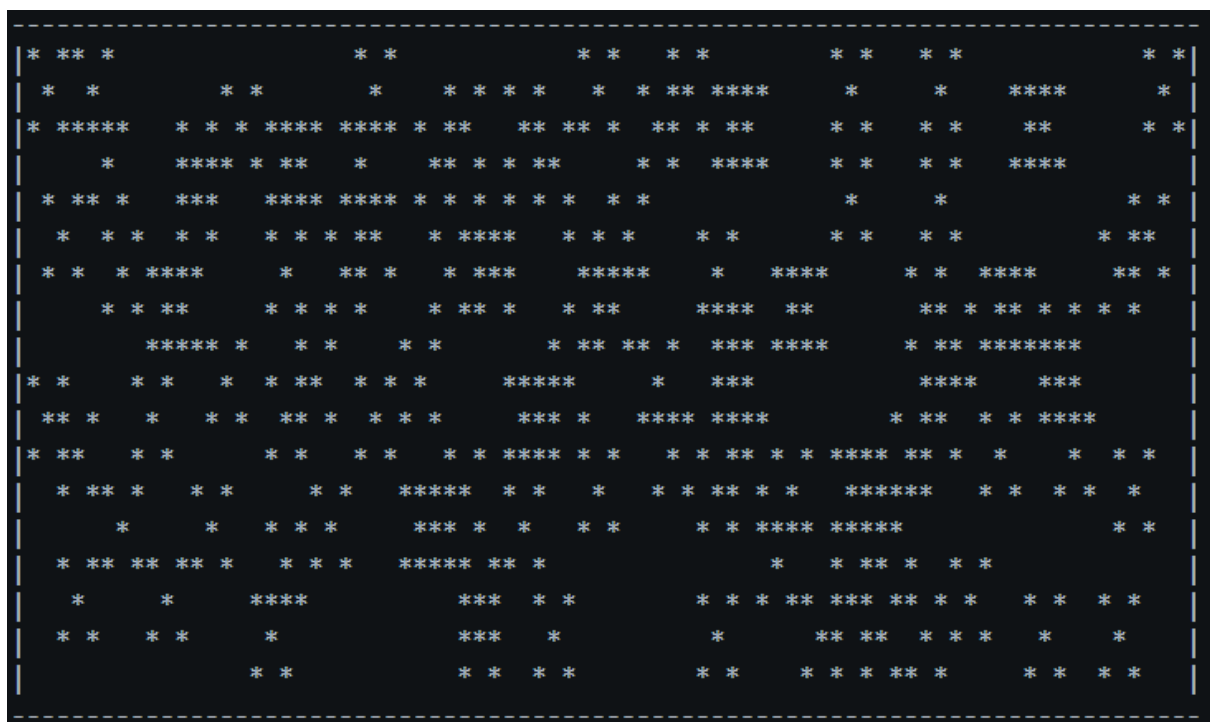


Figura 10.2 – Terceira opção "X", 100 figuras.

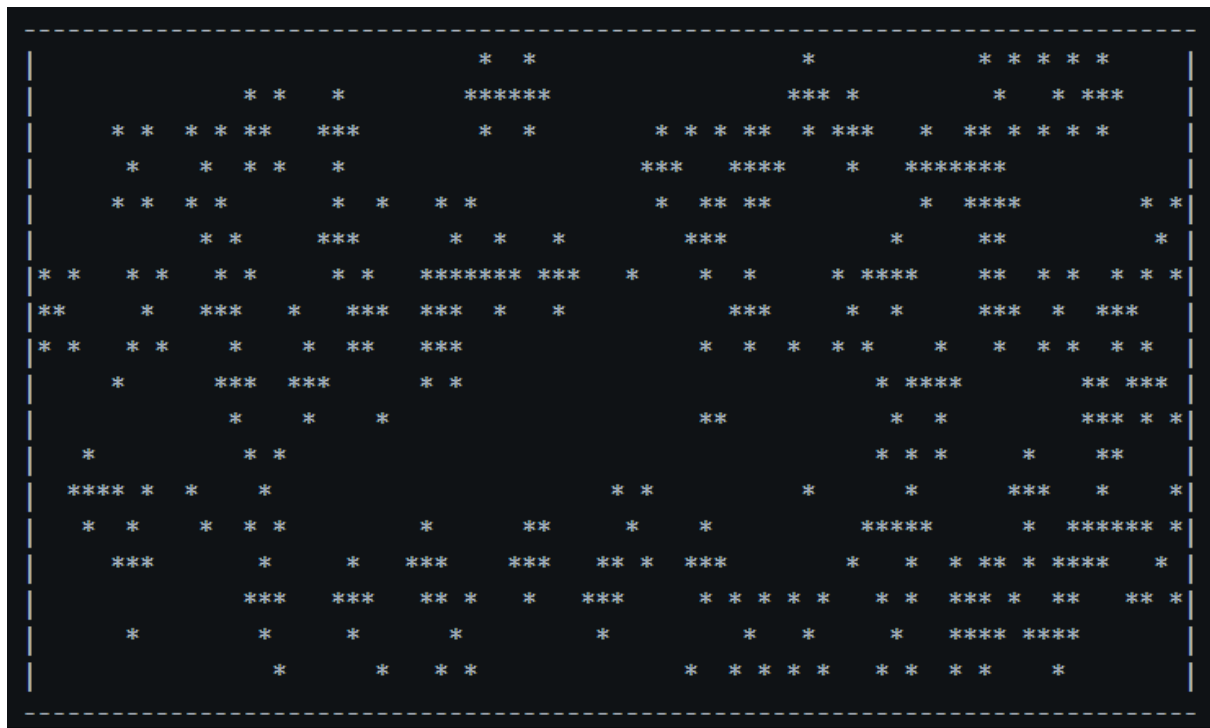


Figura 11.1 – Quarta opção "União aleatória dos 3 primeiros", 100 figuras.

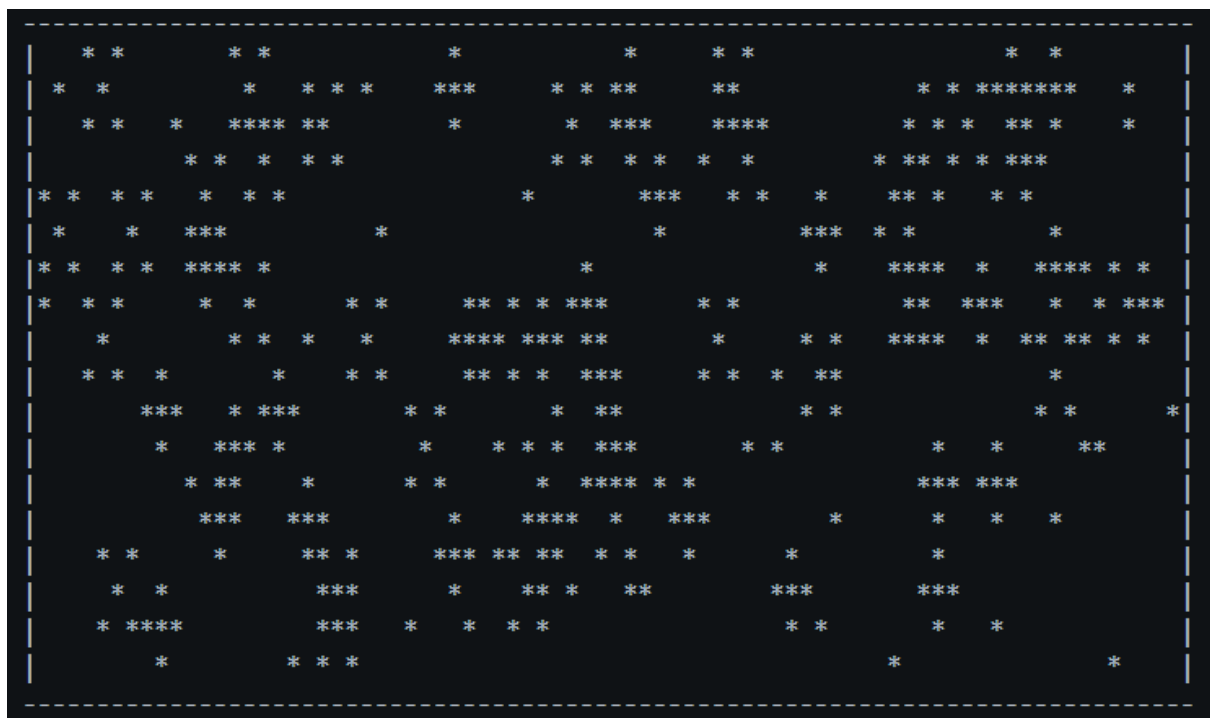


Figura 11.2 – Quarta opção "União aleatória dos 3 primeiros", 100 figuras.

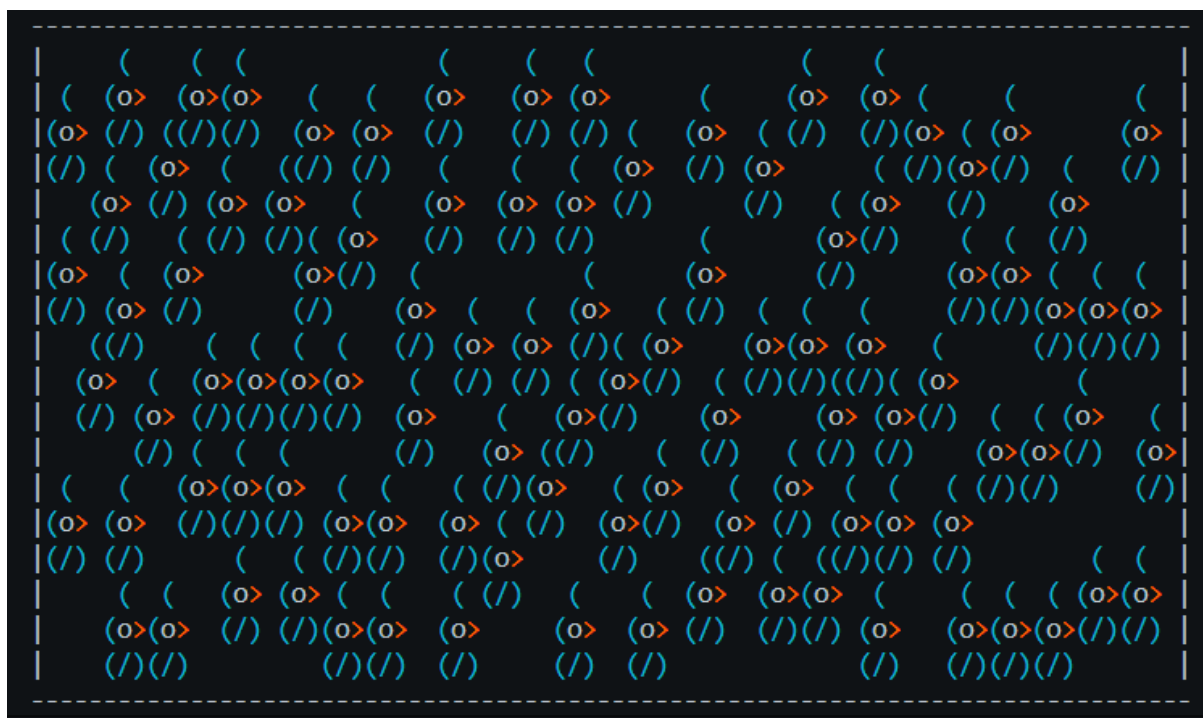


Figura 12.1 – Quinta opção "Calopsitas de uma cor aleatória", 100 figuras.

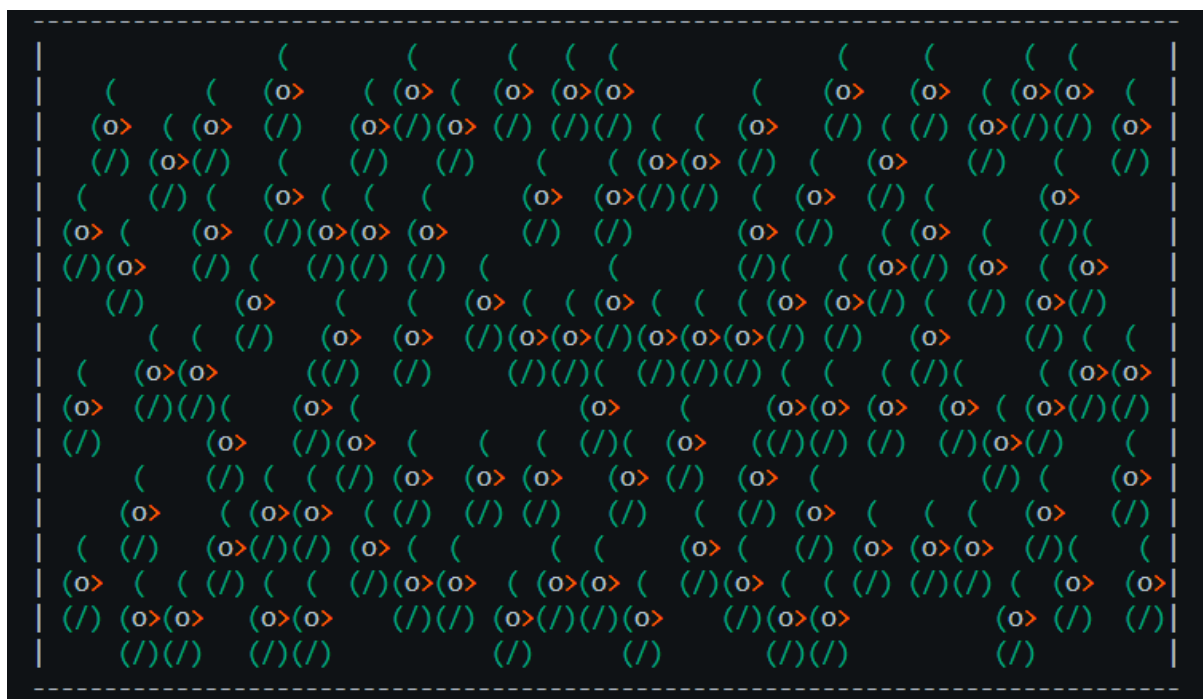


Figura 12.2 – Quinta opção "Calopsitas de uma cor aleatória", 100 figuras.



Figura 12.3 – Quinta opção "Calopsitas de uma cor aleatória", 100 figuras.

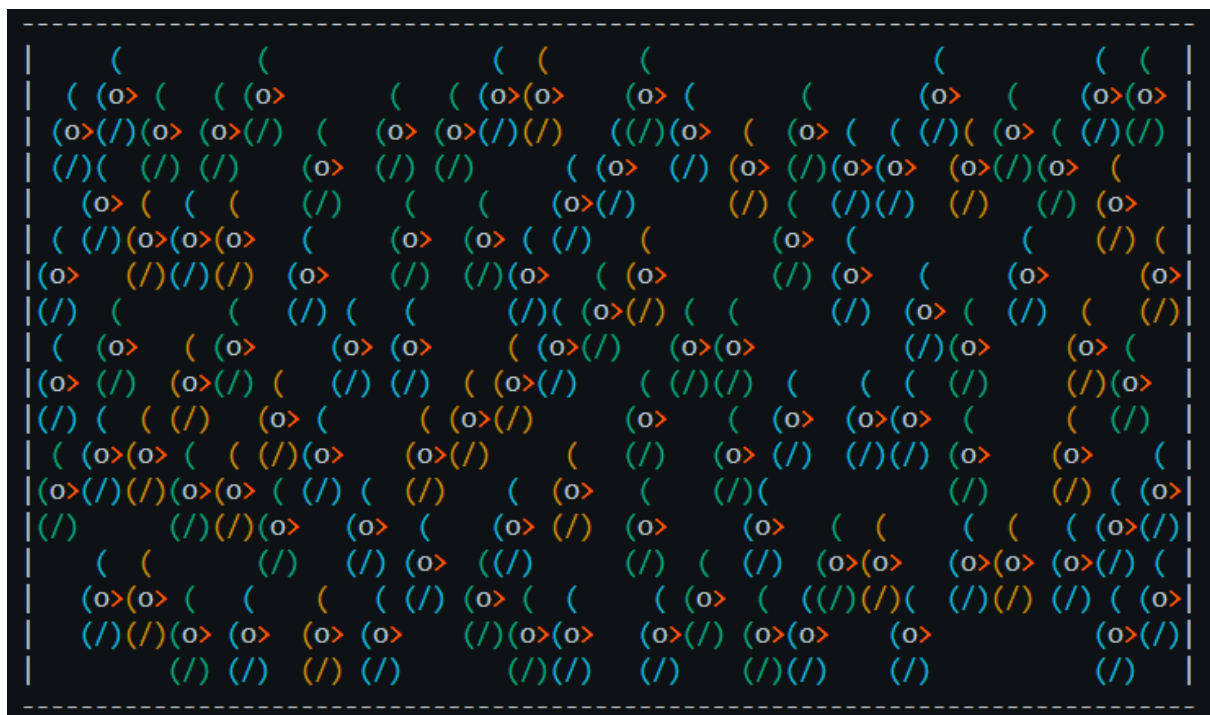


Figura 13 – Sexta opção "Calopsitas de uma cor aleatória cada", 100 figuras.

6. Conclusão

Ao final do projeto, conclui-se que os resultados obtidos com a implementação planejada atenderam de forma satisfatória às demandas estabelecidas. Além disso, ficou evidente que as habilidades de lógica de programação e o domínio da sintaxe da linguagem C foram devidamente lembradas pela aluna. O projeto também incentivou a criatividade e o pensamento crítico, permitindo a criação de um trabalho alinhado ao que foi solicitado.

Em questão de eficiência do código pode se afirmar que há oportunidades de melhoria, porém se trata de um algoritmo que cumpre o que foi solicitado, não há entradas grandes e não é necessário escalabilidade, logo não se torna necessário maior eficiência no código criado.

7. Referências

[1] Color text in terminal applications in UNIX [duplicate]. Disponível em: <<https://stackoverflow.com/questions/3585846/color-text-in-terminal-applications-in-unix/23657072#23657072>>

[2] Inspiração: A large collection of **ASCII art** drawings and other related ASCII art pictures. < <https://www.asciiart.eu/cartoons>>

[3] C progressivo <<https://www.cprogressivo.net/2013/03/Como-gerar-numeros-aleatorios-em-C-com-a-rand-srand-e-seed.html#:~:text=A%20função%20que%20gera%20números,que%20adicionar%20a%20biblioteca%20time.>>