



UNIVERSIDADE FEDERAL DE VIÇOSA - CAMPUS FLORESTAL
HERON FILLIPE

"TRABALHO PRÁTICO - 0"

Trabalho Prático de Projeto e Análise de Algoritmos.

FLORESTAL
2024

Florestal - MG
2024

"TRABALHO PRÁTICO - 0"

Trabalho Prático de Projeto e Análise de Algoritmos.

Heron Fillipe Silveira Santos[4211]

FLORESTAL
2024

Sumário

Introdução	4
Metodologia	4
Desenvolvimento	4
Folder:	4
Makefile:	4
Bibliotecas	5
Funções iniciais:	5
Funções de desenho:	5
Figura criada:	6
Função 'posicaoValida':	6
Função imprimeQuadro:	7
Função:Main	7
Resultado	8
Referências	10

Introdução

Este trabalho tem como objetivo o desenvolvimento de um programa para gerar obras de artes aleatórias. Para isso, o programa deverá respeitar o quadro de limite, de 20 linhas por 80 colunas, sendo na primeira e última linha impresso o símbolo '-' e na primeira e última linha o símbolo '|'.

Metodologia

O trabalho foi realizado em linguagem C de programação utilizando GitHub e VS Code, para facilitar o desenvolvimento em conjunto.

No desenvolvimento do trabalho separamos em etapas, primeiro, desenvolvimento do quadro limite, na segunda, criação dos símbolos já preestabelecidos na documentação e por, na terceira, a função de verificação de posição e por fim, o símbolo de raio (criado pelo aluno).

Desenvolvimento

Folder:

No desenvolvimento deste trabalho foi utilizado apenas a 'main', contendo o menu de opções e realizando todas as operações.

Makefile:

Para execução do código foi criado um makefile simples para facilitar a compilação.

```
#Variáveis de compilador e flags
CC = gcc
CFLAGS = -Wall -g
SRC_DIR = src
OBJ_DIR = obj
BIN_DIR = bin
TARGET = $(BIN_DIR)/programa_arte

#Arquivos fonte e objetos
SRCS = $(wildcard $(SRC_DIR)/*.c)
OBJS = $(SRCS:$(SRC_DIR)/%.c=$(OBJ_DIR)/%.o)

#Regra padrão
all: $(TARGET)

#Regra para compilar o executável final
$(TARGET): $(OBJS) | $(BIN_DIR)
    $(CC) $(CFLAGS) -o $@ $(OBJS)

#Regra para compilar arquivos .c em .o
$(OBJ_DIR)/%.o: $(SRC_DIR)/%.c | $(OBJ_DIR)
    $(CC) $(CFLAGS) -c $< -o $@

#Cria diretórios se não existirem
$(BIN_DIR):
    mkdir -p $(BIN_DIR)

$(OBJ_DIR):
    mkdir -p $(OBJ_DIR)

#Limpeza
clean:
    rm -rf $(OBJ_DIR)/*.o $(TARGET)
```

fig 1.0 Makefile.

Bibliotecas

Para execução, foi incluído 3 bibliotecas:

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
```

Fig 2.0: Bibliotecas.

Sendo a primeira, para utilização das funções de entrada e saída, a segunda para funções auxiliares, como rand e srand, já a última, para manipulação de tempo, usado na geração de números aleatórios.

Funções iniciais:

Na primeira parte do trabalho, foi definido duas constantes e a declaração do quadro, com limites estabelecidos nas orientações do 'TP'. Em seguida, foi criado a função responsável por iniciar o quadro e suas bordas de limite.

Funções de desenho:

Na segunda parte, foram criadas funções para desenhar os tipos de figuras estabelecidas, como asterisco, como e X.

```
// Funções para desenhar as figuras
void desenhaAsterisco(int x, int y) {
    if (posicaoValida(x, y, 1)) {
        quadro[x][y] = '*';
    }
}

void desenhaSoma(int x, int y) {
    if (posicaoValida(x, y, 2)) {
        quadro[x][y] = '*';
        quadro[x + 1][y] = '*';
        quadro[x - 1][y] = '*';
        quadro[x][y + 1] = '*';
        quadro[x][y - 1] = '*';
    }
}

void desenhaX(int x, int y) {
    if (posicaoValida(x, y, 3)) {
        quadro[x][y] = '*';
        quadro[x + 1][y + 1] = '*';
        quadro[x - 1][y - 1] = '*';
        quadro[x + 1][y - 1] = '*';
        quadro[x - 1][y + 1] = '*';
    }
}
```

fig 3.0 Funções de desenho.

Figura criada:

Como pedido nas descrições do trabalho, desenvolvi uma figura diferente, optei por uma no formato de raio.

```
void desenhaRaio(int x, int y) {  
    if (posicaoValida(x, y, 5)) {  
        quadro[x][y] = '*';  
        quadro[x + 1][y+1] = '*';  
        quadro[x + 1][y+2] = '*';  
        quadro[x + 2][y+2] = '*';  
        quadro[x + 2][y+3] = '*';  
        quadro[x + 3][y+2] = '*';  
        quadro[x + 3][y+3] = '*';  
        quadro[x + 3][y+4] = '*';  
        quadro[x + 3][y+5] = '*';  
        quadro[x + 4][y+4] = '*';  
        quadro[x + 4][y+5] = '*';  
        quadro[x + 5][y+5] = '*';  
        quadro[x + 5][y+6] = '*';  
        quadro[x + 6][y+7] = '*';  
    }  
}
```

fig 4.0 Função desenha Raio.

Como mostrado na imagem, ela preenche as posições para desenvolver uma imagem parecida com o raio.

Função 'posicaoValida':

Essa função foi desenvolvida para verificar se uma posição é válida para desenhar uma figura, isso variando de acordo com a figura a ser desenhada.

```
int posicaoValida(int x, int y, int tipoFigura) {  
    //(1x1)  
    if (tipoFigura == 1) {  
        return quadro[x][y] == ' ';  
    }  
    //(3x3)  
    else if (tipoFigura == 2 || tipoFigura == 3) {  
        return x > 0 && x < ALTURA - 1 && y > 0 && y < LARGURA - 1 &&  
            quadro[x][y] == ' ' && quadro[x-1][y] == ' ' && quadro[x+1][y] == ' ' &&  
            quadro[x][y-1] == ' ' && quadro[x][y+1] == ' ' &&  
    }  
    //(7x6)  
    else if (tipoFigura == 5) {  
        return x > 0 && x < ALTURA - 1 && y > 1 && y < LARGURA - 6 &&  
            quadro[x][y] == ' ' &&  
            quadro[x + 1][y + 1] == ' ' &&  
            quadro[x + 1][y + 2] == ' ' &&  
            quadro[x + 2][y + 2] == ' ' &&  
            quadro[x + 2][y + 3] == ' ' &&  
            quadro[x + 3][y + 2] == ' ' &&  
            quadro[x + 3][y + 3] == ' ' &&  
            quadro[x + 3][y + 4] == ' ' &&  
            quadro[x + 3][y + 5] == ' ' &&  
            quadro[x + 4][y + 4] == ' ' &&  
            quadro[x + 4][y + 5] == ' ' &&  
            quadro[x + 5][y + 5] == ' ' &&  
            quadro[x + 5][y + 6] == ' ' &&  
            quadro[x + 6][y + 7] == ' ' &&  
    }  
    return 0;  
}
```

fig 5.0 Função posicaoValida

Como visto na imagem, para todo tipo de figura, ela respeita o limite do quadro. Para o asterisco simples, verifica se o espaço está vazio, para soma e letra x, ela verifica se a posição e suas adjacências estão vazias e para imagem criada (raio), verifica se tem espaço suficiente(7x6)

Função imprimeQuadro:

Para finalizar, foi criada a função responsável por imprimir o resultado do quadro, com as figuras criadas de forma aleatória

```
void imprimeQuadro() {
    for (int i = 0; i < ALTURA; i++) {
        for (int j = 0; j < LARGURA; j++) {
            putchar(quadro[i][j]);
        }
        putchar('\n');
    }
}
```

fig 6.0 Função para imprimir o Quadro.

Função:Main

Na main, está presente o menu de opções e as consequências das escolhas, onde o usuário pode determinar qual tipo de figura ele quer, a quantidade de figuras e por fim, podendo escolher se quer fazer o teste com outro tipo de figura.

```
//menu
printf("PROGRAMA GERADOR DE OBRA DE ARTE:\n");
printf("=====\n");
printf("Escolha o tipo de figura basica a ser usada para criar a obra:\n");
printf("[1] - Asterisco simples.\n");
printf("[2] - Simbolo de soma com asteriscos.\n");
printf("[3] - Letra X com asteriscos.\n");
printf("[4] - Figuras aleatorias.\n");
printf("[5] - Desenho de Raio\n");
printf("Digite o tipo de figura basica desejada: ");
scanf("%d", &tipo);

printf("Digite a quantidade de figuras (menor ou igual a zero para aleatorio): ");
scanf("%d", &quantidade);

if (quantidade <= 0) {
    quantidade = rand() % 100 + 1;
    printf("\nValor fora do limite!\n Novo valor atribuido é %i\n", quantidade);
} else if (quantidade > 100) {
    quantidade = 100;
    printf("\nValor fora do limite!\n Novo valor atribuido é %i\n", quantidade);
}
```

fig 7.0 Menu de opções.

```

iniciaQuadro();

for (int i = 0; i < quantidade; i++) {
    int x, y;
    int figuraAleatoria = tipo;

    //tipo for 4, escolhe aleatoriamente entre 1, 2, e 3
    if (tipo == 4) {
        figuraAleatoria = rand() % 3 + 1;
    }

    //Gera novas coordenadas até encontrar uma posição válida
    do {
        x = rand() % (ALTURA - 2) + 1;
        y = rand() % (LARGURA - 2) + 1;
    } while (!posicaoValida(x, y, figuraAleatoria));

    //Desenha a figura
    if (figuraAleatoria == 1) {
        desenhaAsterisco(x, y);
    } else if (figuraAleatoria == 2) {
        desenhaSoma(x, y);
    } else if (figuraAleatoria == 3) {
        desenhaX(x, y);
    } else if (figuraAleatoria == 5) {
        desenhaRaio(x, y);
    }
}

imprimeQuadro();

printf("Deseja gerar um novo quadro com os mesmos valores? (s/n): ");
scanf(" %c", &opcao);

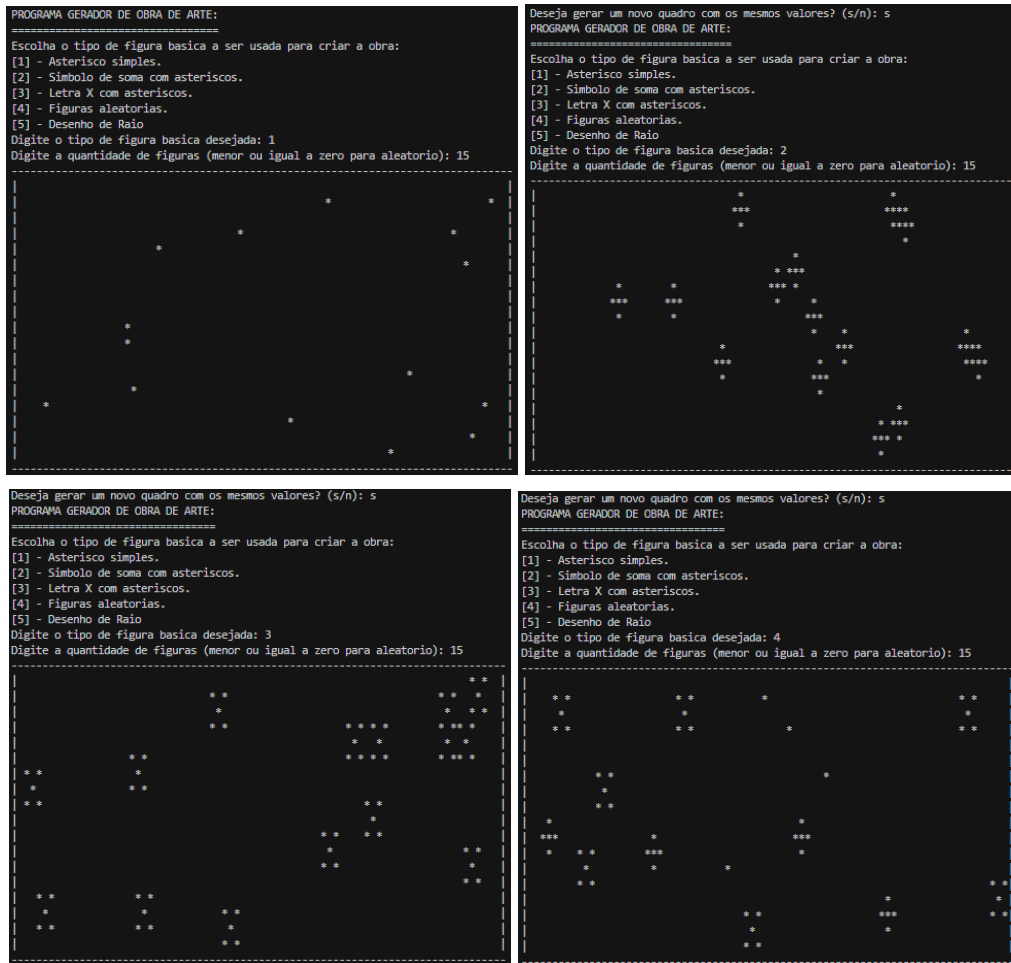
```

fig 8.0 Criação do quadro

Como pode ser visto na imagem acima, nessa parte do código é criado o quadro com as figuras obtendo as coordenadas aleatórias para preencher o espaço vazio do quadro,

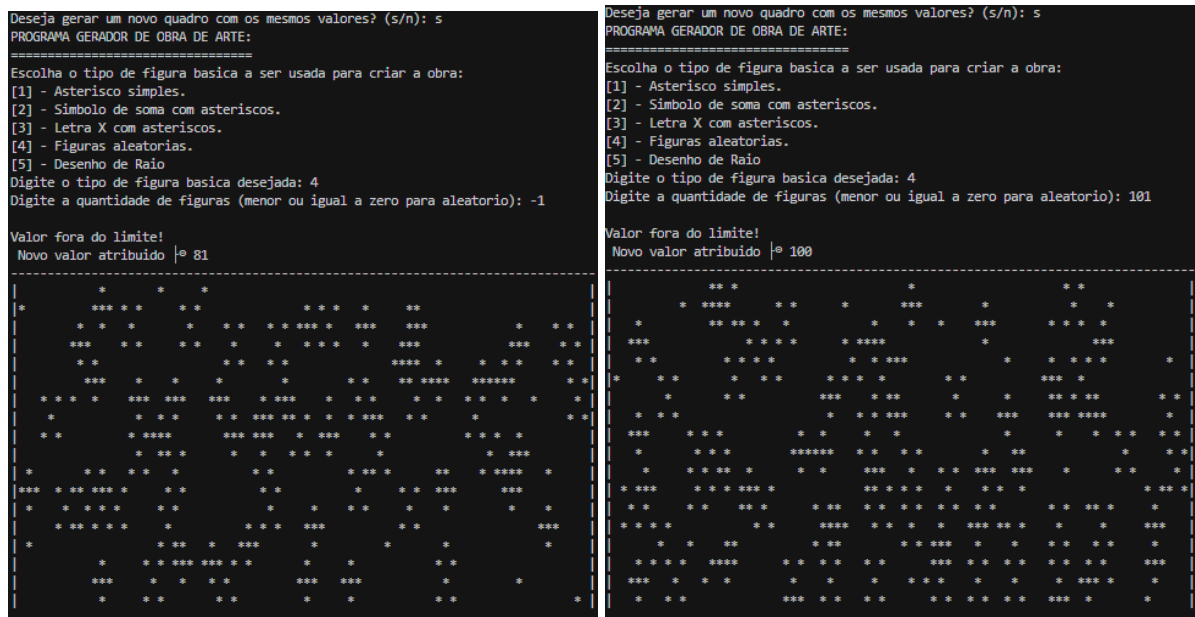
Resultado

Como pedido no trabalho, executei o código e ele nos traz o menu de opções, a seguir será mostrado o que cada opção realiza.



figs 9.0 Teste de cada opção com a quantidade de 15.

Como pedido no trabalho, deveria ser adicionado opção de verificação de valores fora do intervalo 1 a 100.



figs 10.0 Teste de intervalo.

E por fim, teste com a figura criada.

Deseja gerar um novo quadro com os mesmos valores? (s/n): s

PROGRAMA GERADOR DE OBRA DE ARTE:

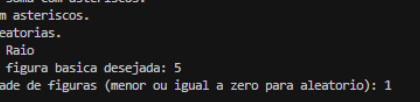
=====

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] - Desenho de Raio

Digite o tipo de figura basica desejada: 5

Digite a quantidade de figuras (menor ou igual a zero para aleatorio): 1



Deseja gerar um novo quadro com os mesmos valores? (s/n): s

PROGRAMA GERADOR DE OBRA DE ARTE:

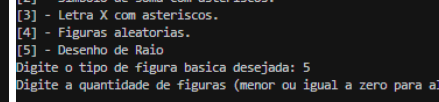
=====

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] - Desenho de Raio

Digite o tipo de figura basica desejada: 5

Digite a quantidade de figuras (menor ou igual a zero para aleatorio): 15



figs 11.0 Teste da figura criada.

Referências

<https://code.visualstudio.com/>

<https://github.com/HeronFillipe/TP0PAA.git>