Universidade Federal de Viçosa - *Campus* Florestal
Bacharelado em Ciência da Computação
CCF 330 - Projeto e Análise de Algoritmos
Prof. Daniel Mendes Barbosa

Trabalho Prático 0

Programa Gerador de Obras de Arte

Samuel Jhonata S. Tavares 2282

Florestal - MG
2018

Sumário

1 Introdução

Este trabalho tem por objetivo criar um programa gerador de "obras de arte"de forma aleatória, na linguagem *C*.

O capítulo 2 deste trabalho vem mostrar o seu desenvolvimento, começando pela seção 2.1, com algumas considerações gerais, passando pela seção 2.2, mostrando alguns detalhes de como foi sua implementação, e logo a seguir, na seção 2.3 tem-se uma breve execução do programa, apresentada em algumas capturas de tela.

Por fim, no capítulo 3 é apresentada uma breve conclusão do trabalho.

2 Desenvolvimento

2.1 Considerações Gerais

Para o desenvolvimento deste trabalho, foi utilizada a Linguagem C, no Sistema Operacional Windows, com a IDE Netbeans v8.2. Para controle de versão, foi utilizado o *Git Hub*.

O programa implementado consiste em um gerador de obras de arte aleatória, que deve receber do usuário o tipo de figura (formada por asteriscos) e a quantidade a ser gerada, podendo variar entre 1 e 100 figuras, num quadro 20x80.

Uma abordagem inicial foi utilizar da recursão para a geração das figuras, porém, este método se mostrou muito ineficiente, uma vez que, para valores não muito grandes de figuras, já não era possível executar.

A figura escolhida para ser implementada pelo aluno foi a letra "T", como na Figura 1 abaixo:

Figura 1 – Figura escolhida pelo aluno

2.2 Implementação

Entendido o problema, deu-se início à implementação. Buscando uma implementação modular, criou-se os arquivos seguintes arquivos:

2.2.1 *ObraArte.h*

"ObraArte.h", mostrado na Figura 2, com algumas definições e os protótipos das funções implementadas em ***"ObraArte.c"***

Figura 2 – Arquivo *"ObraArte.h"*

```
#ifndef OBRAARTE_H_INCLUDED
#define OBRAARTE_H_INCLUDED

#define ALTURA 20
#define LARGURA 80

#define ASTERISCO 1
#define SOMA 2
#define LX 3
#define LT 4

void criarQuadroVazio(char quadro[][LARGURA]);
void mostrarQuadro(char quadro[][LARGURA]);
void insereAsterisco(char quadro[][LARGURA], int qtd);
void insereSoma(char quadro[][LARGURA], int qtd);
void insereX(char quadro[][LARGURA], int qtd);
void insereAleatorio(char quadro[][LARGURA], int qtd);
void insereOutra(char quadro[][LARGURA], int qtd);
int verificaVazio(char quadro[][LARGURA], int x, int y);
int geraNumero(int max);
void contaFiguraADD(int totalf);


#endif // OBRAARTE_H_INCLUDED
```

2.2.2 *ObraArte.c*

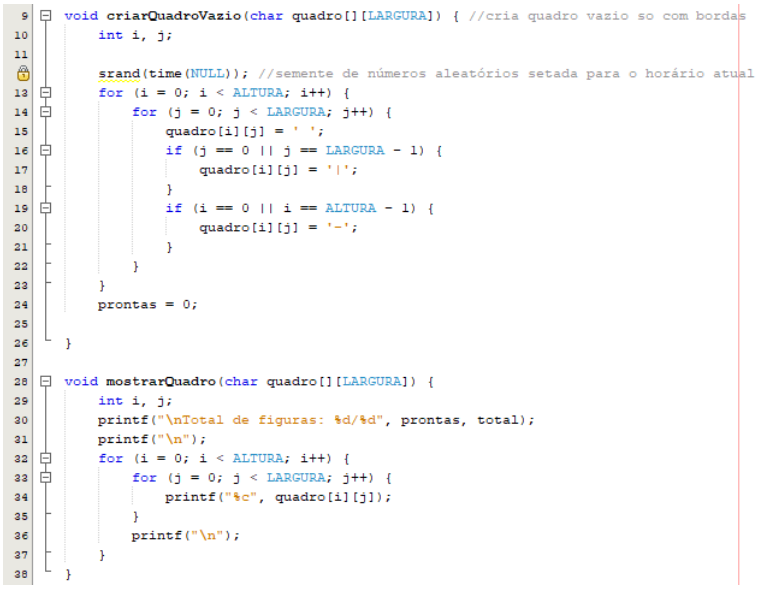
O arquivo ***"ObraArte.c"***, com suas primeiras linhas mostradas na Figura 3, define duas variáveis globais, *"prontas"* e *"total"*, utilizadas para contar as figuras. Nesse arquivo também estão as implementações dos seguintes procedimentos e funções que serão descritos logo a seguir.

Figura 3 – Arquivo *"ObraArte.c"*

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

#include "ObraArte.h"
int prontas, total;
```

O procedimento ***"criarQuadroVazio()"***, que cria um quadro de 20 linhas por 80 colunas, ou seja, uma matriz de caracteres 20x80, onde as linhas e colunas mais externas marcam a moldura e as posições internas, começam com um espaço em branco. E o procedimento ***"mostrarQuadro()"***, que mostra o quadro na tela, percorrendo toda a matriz., ambos mostrados na Figura 4.

Figura 4 – Procedimentos *"criarQuadroVazio()"* e *"mostrarQuadro()"*



```
9  void criarQuadroVazio(char quadro[][LARGURA]) { //cria quadro vazio so com bordas
10     int i, j;
11
12     srand(time(NULL)); //semente de números aleatórios setada para o horário atual
13     for (i = 0; i < ALTURA; i++) {
14         for (j = 0; j < LARGURA; j++) {
15             quadro[i][j] = ' ';
16             if (j == 0 || j == LARGURA - 1) {
17                 quadro[i][j] = '|';
18             }
19             if (i == 0 || i == ALTURA - 1) {
20                 quadro[i][j] = '-';
21             }
22         }
23     }
24     prontas = 0;
25
26 }
27
28 void mostrarQuadro(char quadro[][LARGURA]) {
29     int i, j;
30     printf("\nTotal de figuras: %d/%d", prontas, total);
31     printf("\n");
32     for (i = 0; i < ALTURA; i++) {
33         for (j = 0; j < LARGURA; j++) {
34             printf("%c", quadro[i][j]);
35         }
36         printf("\n");
37     }
38 }
```

Os procedimentos ***"insereSoma()"***, ***"insereX()"*** e ***"insereAleatorio()"***, que inserem no quadro os símbolos descritos na especificação deste trabalho e ***"insereOutra()"***, que insere asteriscos em forma de "T", sendo a imagem escolhida pelo aluno. Como parâmetro, é enviada a quantidade de figuras a serem geradas. Cada uma delas sorteia uma posição, composta por linha e coluna, para tentar inserir a figura correspondente, verificando se as respectivas posições da matriz estão vazias (preenchida com espaço vazio) e, caso contrário, sorteia outros números até conseguir um lugar totalmente vazio. Uma particularidade de ***"insereAleatorio()"*** é que é sorteado um número aleatoriamente para indicar qual figura será inserida a cada nova inserção. É possível ver esses procedimentos na Figura 5.

Figura 5 – Procedimentos *"insereAsterisco()"*, *"insereSoma()"*, *"insereX()"*, *"insereAleatorio()"* e *"insereOutra()"*

```
void insereAsterisco(char quadro[][LARGURA], int qtd) {
    int i;
    int x, y;

    for (i = 0; i < qtd; i++) {
        x = geraNumero(ALTURA);
        y = geraNumero(LARGURA);
        if (verificaVazio(quadro, x, y)) { //se as posições estão vazias
            quadro[x][y] = '*';
            contaFiguraADD(qtd);
        } else { //se as posições não estão vazias
            i--;
        }
    }
}

void insereSoma(char quadro[][LARGURA], int qtd) {
    int i;
    int x, y;

    for (i = 0; i < qtd; i++) {
        x = geraNumero(ALTURA);
        y = geraNumero(LARGURA);
        if (verificaVazio(quadro, x, y) && verificaVazio(quadro, x, y + 1) &&
                verificaVazio(quadro, x - 1, y) && verificaVazio(quadro, x + 1, y) &&
                verificaVazio(quadro, x, y - 1)) { //se as posições estão vazias
            quadro[x][y + 1] = '*';
            quadro[x - 1][y] = '*';
            quadro[x][y] = '*';
            quadro[x + 1][y] = '*';
            quadro[x][y - 1] = '*';
            contaFiguraADD(qtd);
        } else { //se as posições não estão vazias
            i--;
        }
    }
}
void insereX(char quadro[][LARGURA], int qtd) {
    int i;
    int x, y;

    for (i = 0; i < qtd; i++) {
        x = geraNumero(ALTURA);
        y = geraNumero(LARGURA);
        if (verificaVazio(quadro, x, y) && verificaVazio(quadro, x - 1, y - 1) &&
                verificaVazio(quadro, x - 1, y + 1) && verificaVazio(quadro, x + 1, y - 1) &&
                verificaVazio(quadro, x + 1, y + 1)) { //se as posições estão vazias
            quadro[x][y] = '*';
            quadro[x - 1][y - 1] = '*';
            quadro[x - 1][y + 1] = '*';
            quadro[x + 1][y - 1] = '*';
            quadro[x + 1][y + 1] = '*';
            contaFiguraADD(qtd);
        } else { //se as posições não estão vazias
            i--;
        }
    }

}
```

```
void insereOutra(char quadro[][LARGURA], int qtd) {
    int i;
    int x, y;

    for (i = 0; i < qtd; i++) {
        x = geraNumero(ALTURA);
        y = geraNumero(LARGURA);
        if (verificaVazio(quadro, x, y) && verificaVazio(quadro, x - 1, y + 1) &&
                verificaVazio(quadro, x - 1, y - 1) && verificaVazio(quadro, x + 1, y) &&
                verificaVazio(quadro, x - 1, y)) { //se as posições estão vazias
            quadro[x - 1][y + 1] = '*';
            quadro[x - 1][y - 1] = '*';
            quadro[x + 1][y] = '*';
            quadro[x][y] = '*';
            quadro[x - 1][y] = '*';
            contaFiguraADD(qtd);
        } else { //se as posições não estão vazias
            i--;
        }
    }

}
void insereAleatorio(char quadro[][LARGURA], int qtd) {
    int i, num;
    for (i = 0; i < qtd; i++) {
        num = geraNumero(3) + 1; //sorteia o tipo de figura
        switch (num) {
            case ASTERISCO:
                insereAsterisco(quadro, 1);
                break;
            case SOMA:
                insereSoma(quadro, 1);
                break;
            case LX:
                insereX(quadro, 1);
                break;
            default:
                i--;
                break;
        }
    }
    total = qtd; //total de figuras geradas
}
```

A função ***"verificaVazio()"*** é responsável por verificar se uma determinada posição, dada linha e coluna, está disponível para receber um asterisco, retornando 1 caso positivo e 0 caso não seja possível.

A função ***"geraNumero()"*** retorna um número aleatório entre 0 e um número máximo, passado por parâmetro.

O procedimento ***"contaFiguraADD()"*** é utilizado para controlar a quantidade de figuras que já foram inseridas.

Na Figura 6 são mostradas as funções ***"verificaVazio()"***, ***"geraNumero()"*** e ***"contaFiguraADD()"***.

Figura 6 – Procedimentos *"criarQuadroVazio()"* e *"mostrarQuadro()"*

```
146 int verificaVazio(char quadro[][LARGURA], int x, int y) { //olha se a posição está vazia
147     if (quadro[x][y] == ' ') {
148         return 1;
149     }
150     return 0;
151
152 }
153
154 int geraNumero(int max) { // gera um número aleatório entre 0 e max
155     return rand() % (max);
156 }
157
158 void contaFiguraADD(int totalf) { //adiciona mais uma figura pronta
159     prontas++;
160     total = totalf;
161 }
```

2.2.3 *Main.c*

O arquivo principal *"main.c"* é mostrado na Figura 7 e nele é implementado o menu principal do programa, responsável por capturar as entradas do usuário e invocar os métodos responsáveis por criar o quadro, inserir as figuras e imprimir o quadro gerado.

Figura 7 – Arquivo *"main.c"*

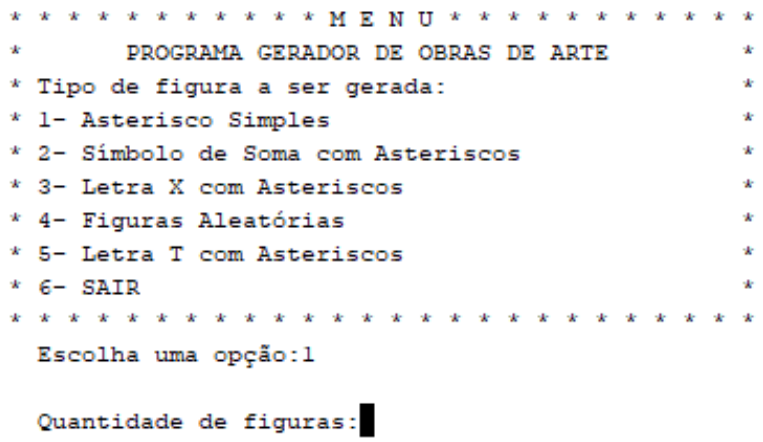
```
#include <stdio.h>
#include <stdlib.h>
#include "ObraArte.h"

int main() {
    char quadro[ALTURA][LARGURA];
    int opcao = 0, qtd;

    while (opcao != 6) {
        printf("            * * * * * * * * * * * * M E N U * * * * * * * * * * * * *");
        printf("\n             *       PROGRAMA GERADOR DE OBRAS DE ARTE          *");
        printf("\n             * Tipo de figura a ser gerada:                     *");
        printf("\n             * 1- Asterisco Simples                             *");
        printf("\n             * 2- Símbolo de Soma com Asteriscos                *");
        printf("\n             * 3- Letra X com Asteriscos                        *");
        printf("\n             * 4- Figuras Aleatórias                            *");
        printf("\n             * 5- Letra T com Asteriscos                        *");
        printf("\n             * 6- SAIR                                          *");
        printf("\n             * * * * * * * * * * * * * * * * * * * * * * * * * *");


        do {
            printf("\n              Escolha uma opção:");
            fflush(stdout);
            scanf("%d", &opcao);
            fflush(stdout);
        } while (opcao < 1 || opcao > 6);

        if (opcao == 6) {
            printf("Programa encerrado!");
            break;
        }

        printf("\n              Quantidade de figuras:");
        fflush(stdout);
        scanf("%d", &qtd);
        fflush(stdout);

        if (qtd <= 0) {
            qtd = geraNumero(99) + 1;
            printf("\n                    - Serão geradas %d figuras", qtd);
        } else if (qtd > 100) {
            qtd = 100;
        }

        criarQuadroVazio(quadro);
        switch (opcao) {
            case 1:
                insereAsterisco(quadro, qtd);
                break;
            case 2:
                insereSoma(quadro, qtd);
                break;
            case 3:
                insereX(quadro, qtd);
                break;
            case 4:
                insereAleatorio(quadro, qtd);
                break;
            case 5:
                insereOutra(quadro, qtd);
                break;
        }
        mostrarQuadro(quadro);
    }

    return 0;
}
```

2.3 Execução

Após toda a implementação e alguns testes, chegou-se na versão final do programa. A seguir, serão mostradas algumas capturas de tela para melhor ilustrar os resultados obtidos.

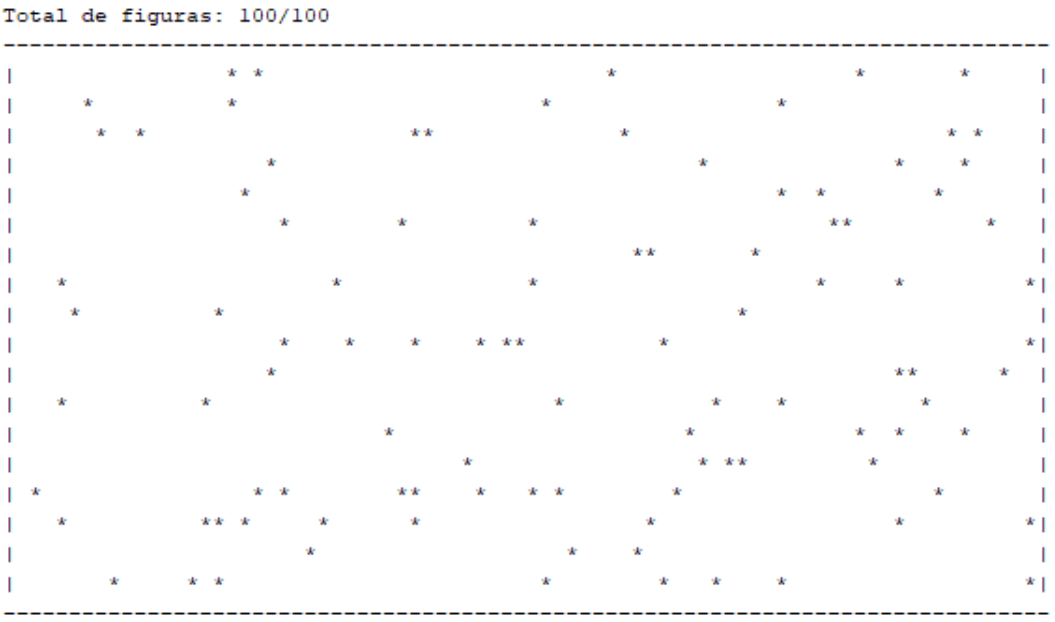
Na figura 8, é apresentado o menu principal do programa, onde é possível ao usuário escolher as figuras que quer que o programa gere.

Figura 8 – Captura do menu principal



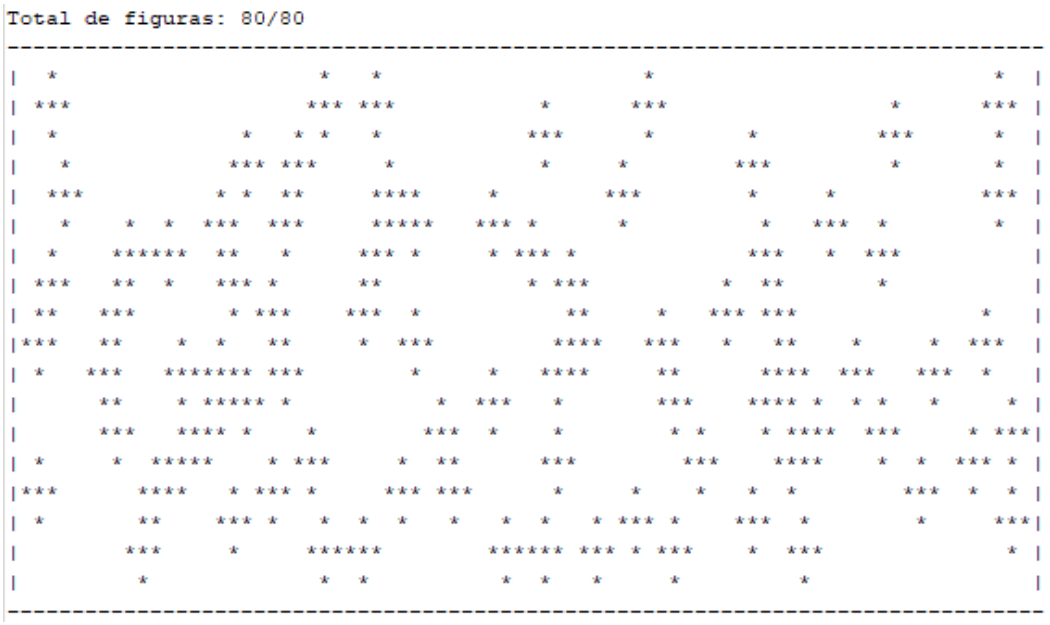
Na figura 9, é apresentada a geração de uma obra de arte com 100 asteriscos, entrada pelo usuário.

Figura 9 – Captura da geração de figuras com 100 asteriscos



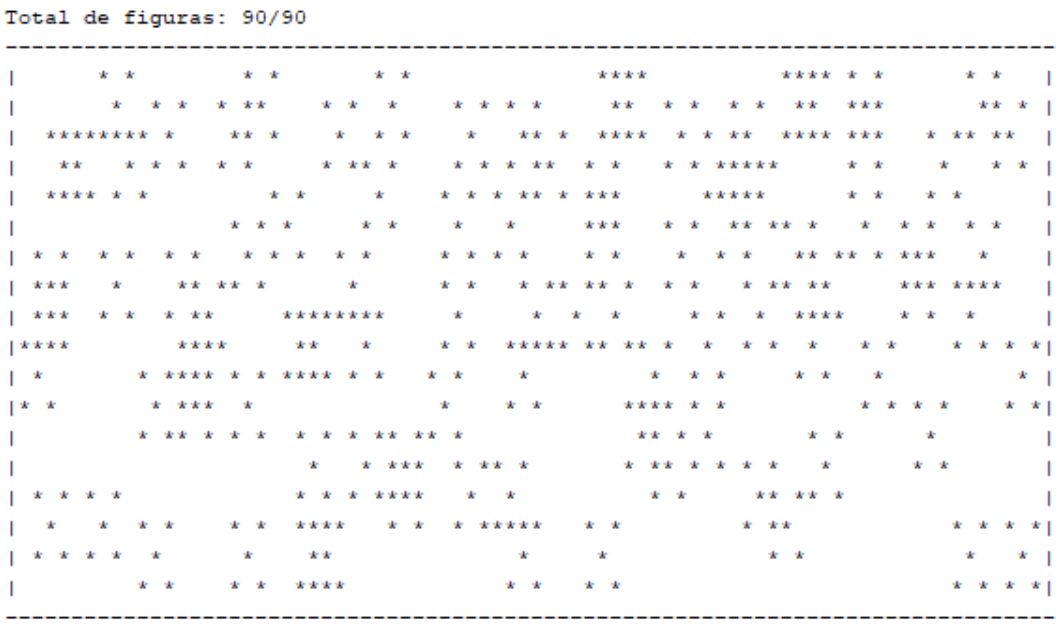
Na figura 10, é apresentada a geração de uma obra de arte com 80 símbolos de soma, formadas com asteriscos, entrada pelo usuário.

Figura 10 – Captura da geração de figuras com 80 símbolos de soma



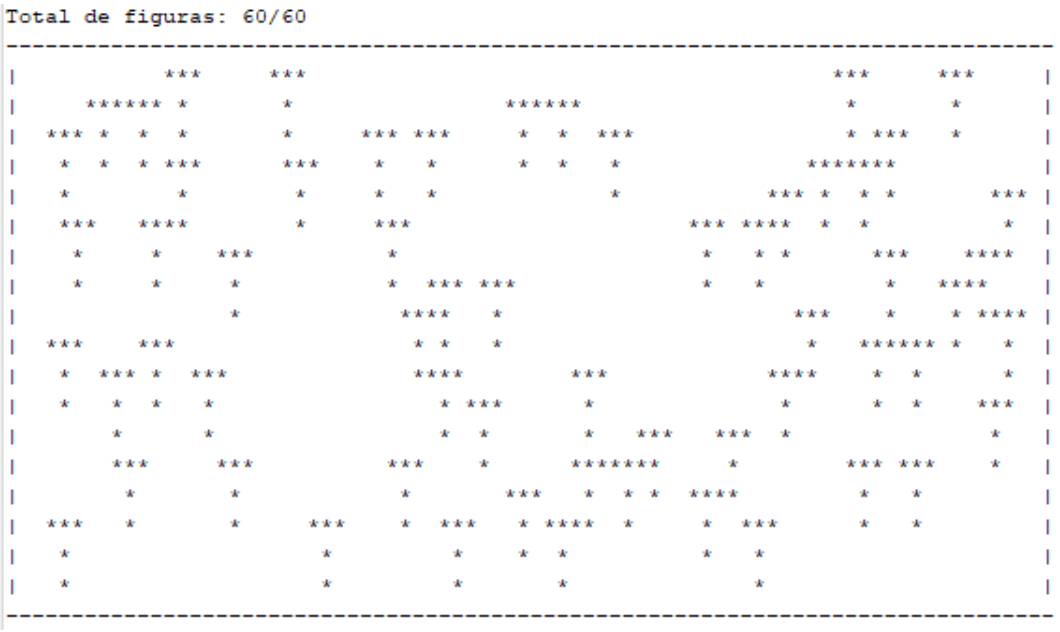
Na figura 11, é apresentada a geração de uma obra de arte com 90 figuras X, formadas com asteriscos, entrada pelo usuário.

Figura 11 – Captura da geração de figuras com 90 figuras X



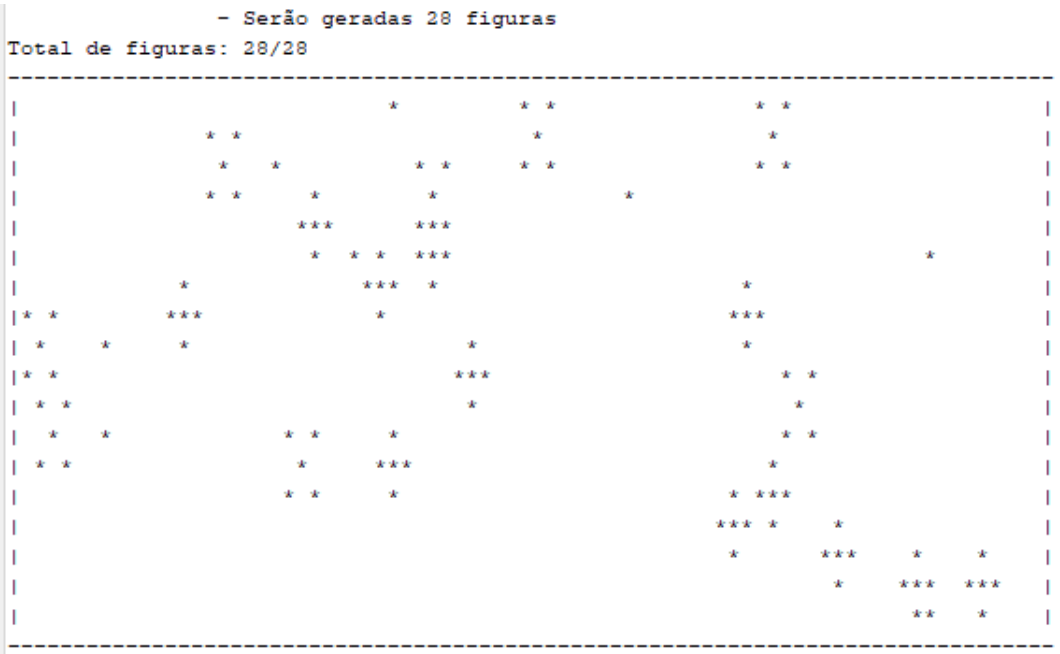
Na figura 12, é apresentada a geração de uma obra de arte com 60 figuras T, formadas com asteriscos, entrada pelo usuário.

Figura 12 – Captura da geração de figuras com 60 T



Na figura 13, é apresentada a geração de uma obra de arte com 28 figuras T, formadas com figuras aleatórias. A quantidade de figuras foi gerada de forma aleatória, uma vez que o usuário entrou com um valor menor que 1. Caso um valor maior que 100 seja dado por entrada, a quantidade de figuras geradas será sempre 100.

Figura 13 – Captura da geração de figuras com 28 figuras escolhidas de forma aleatória



3 Conclusão

Com este trabalho, foi possível implementar e executar um arquivo gerador de obras de arte a partir da colocação aleatória das figuras asterisco simples, símbolo de soma, letra X e letra T, proporcionando configurar o ambiente de desenvolvimento para a Linguagem C.