Universidade Federal de Viçosa - Campus Florestal

Trabalho 00 da Disciplina de Projeto e Análise de Algoritmos

Programa gerador de obra de arte

Isabella Menezes Ramos 3474

Professor: Daniel Mendes Barbosa

Florestal - MG

19 de Fevereiro de 2021

Isabella Menezes Ramos 3474

Trabalho 00 da Disciplina de Projeto e Análise de Algoritmos

Primeiro trabalho prático da disciplina Projeto e Análise de Algoritmos. O trabalho tem o intuito de criar um programa gerador de obras de arte.

Professor: Daniel Mendes Barbosa

Florestal - MG
19 de Fevereiro de 2021

Sumário

1 Introdução

O trabalho proposto tem o objetivo de criar um programa para gerar obras de arte aleatórias através de um quadro de 20 linhas por 80 colunas, onde na primeira e última linhas deverá ser impresso o símbolo '-' em todas as 80 colunas e na primeira e última coluna deverá ser impresso o símbolo '|' em todas as linhas, com exceção da primeira e última que já irão conter o símbolo '-'.

O programa deverá ter um menu onde o usuário deve escolher qual tipo de figura básica irá utilizar em seu quadro. Poderá escolher apenas uma das três, uma mistura aleatória das três ou uma obra de arte criada pelo aluno, sendo que, logo em seguida, deverá escolher quantas figuras serão utilizadas para a geração do quadro. Se o usuário digitar um número menor ou igual a zero, será gerado um número de figuras aleatório entre 1 e 100. Se o usuário digitar um número maior do que 100, será considerado o número 100.

Nos capítulos seguintes, iremos detalhar o desenvolvimento do trabalho. No capítulo 2, foi detalhado como foi feito o programa, detalhando as funções principais mostrando exemplos das saídas da execução do programa. No capítulo 3, mostra como executar o trabalho e capturas de tela da execução do programa. Concluímos o trabalho no capítulo 4.

2 Desenvolvimento

O código está dividido da seguinte forma, o arquivo funcoes.c é onde se encontram as funções que fazem o programa funcionar de fato e o arquivo menu.c se encontra funções com o intuito de “encapsular” partes do código do menu principal.

```
void inicializaMatriz(structMatriz *matriz);
void imprimeMatriz(structMatriz matriz);
void criaAsterisco(structMatriz *matriz, int quantidade);
void criaSoma(structMatriz *matriz, int quantidade);
void criaLetraX(structMatriz *matriz, int quantidade);
void criaAleatorias(structMatriz *matriz, int quantidade);
void criaObraAluno(structMatriz *matriz, int quantidade);
void criaObraSorrindo(structMatriz *matriz, int quantidade);
void criaObraPiscando(structMatriz *matriz, int quantidade);
void criaObraDormindo(structMatriz *matriz, int quantidade);
void criaObraSurpresa(structMatriz *matriz, int quantidade);
int verificaQuantidade(int quantidade);
int verificaQuantidade2(int quantidade);
```

Figura 01. Funções presentes no funcoes.c.

```
void menuPrincipal();
int verificaMenorQueZero(int quantidade);
```

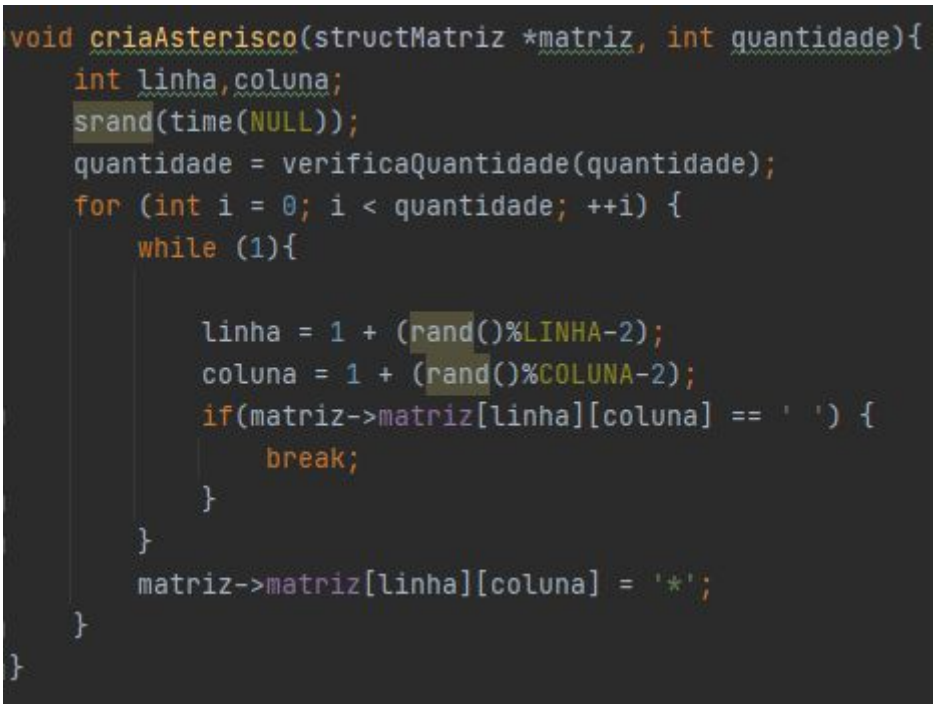
Figura 02. Funções presentes no menu.c.

Para criar o quadro que será colocado as obras de artes, foi criado uma matriz, e foi utilizado duas estruturas de repetição, uma dentro da outra, para preencher as bordas da matriz ou com ‘|’ para as laterais do quadro, ou ‘-’ para a parte superior e inferior do quadro. Os outros espaços da matriz disponíveis, que não foi preenchida com os símbolos ditos anteriormente, preenchemos com um espaço em branco. A criação do quadro foi feita na função *inicializaMatriz*.

O funcionamento de criação das figuras em posições aleatórias são semelhantes. Para que a figura que eu desejo criar esteja em uma posição aleatória dentro do quadro, utilizamos a função **rand()** que gera números entre 0 e RAND_MAX, onde esse RAND_MAX é um valor que podemos definir.

E para garantir que a figura que eu quero inserir não sobreponha outra que já esteja no quadro, verificamos se aquela posição da matriz está vazia, caso esteja,

podemos inserir a figura. No caso das figuras que ocupam mais de um espaço na matriz, temos que fazer essa verificação em todos os espaços, se em caso um deles não estiver disponível, terá que ser gerado aleatoriamente outra coordenada para aquela figura.



```
void criaAsterisco(structMatriz *matriz, int quantidade){
    int linha,coluna;
    srand(time(NULL));
    quantidade = verificaQuantidade(quantidade);
    for (int i = 0; i < quantidade; ++i) {
        while (1){

            linha = 1 + (rand()%LINHA-2);
            coluna = 1 + (rand()%COLUNA-2);
            if(matriz->matriz[linha][coluna] == ' ') {
                break;
            }
        }
        matriz->matriz[linha][coluna] = '*';
    }
}
```

Figura 03. Função *criaAsterisco().*

Na figura acima temos uma das funções de criação de figuras do nosso programa, perceba que foi utilizado também a função **srand()**, com o intuito de que a cada execução do programa seja criado as figuras em lugares diferentes do quadro.

Além das funções principais de criação, temos também funções auxiliares denominadas de *verificaQuantidade* e *verificaQuantidade2* presentes em funcoes.c e *verificaMenorQueZero* presente em menu.c, com o intuito de verificar as exceções descritas na especificação do trabalho. A função *verificaQuantidade* verifica se o número é maior que 100, e caso for, ele retorna o valor 100. A função *verificaQuantidade2* é utilizada nas funções de criação de obra aleatória e na obra feita pelo aluno, e tem o intuito de verificar que a quantidade de figuras não seja negativa. Já a função *verificaMenorQueZero*, presente no main.c, verifica outra exceção dita na especificação, de que se a quantidade de figuras dita pelo usuário for menor que zero, será gerado uma quantidade de figuras aleatórias no intervalo de 1 a 100.

A obra de arte escolhida pelo aluno foi um conjunto de Kaomojis criados aleatoriamente. Kaomoji é o nome dado ao estilo de emoticon japonês, que representa expressões faciais, criado por meio de uma mistura de caracteres japoneses, escritas ocidentais e caracteres de byte duplo. Mas com testes, descobri que caracteres de byte duplo, muitas vezes utilizados em Kaomojis, não eram aceitos pela linguagem, então a solução foi utilizar caracteres que a linguagem conseguia exibir e armazenar em cada espaço da matriz.

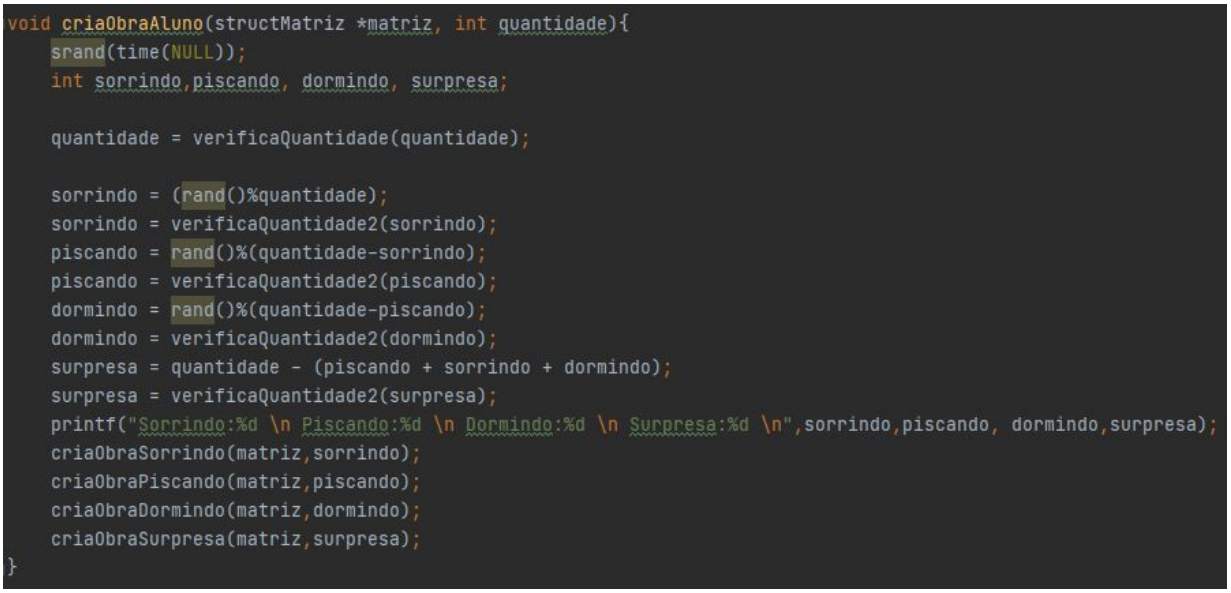
Figura 04. Captura de tela da saída da opção 5 do menu do programa.

No total, temos quatro tipos de kaomojis no programa, como são gerados aleatoriamente, dependendo da quantidade de figuras, pode ser que não apareça os quatro. Abaixo temos uma legenda de cada figura:

Sorrindo	(*^-^*)
Piscando	(^-~)
Dormindo	(-_-)zzZ
Surpresa	<(OoO)>

Tabela 01. Kaomojis utilizados no programa

Para cada kaomoji foi criado uma função separada, e todos eles são chamados dentro da função *criaObraAluno* que está na figura abaixo.



```
void criaObraAluno(structMatriz *matriz, int quantidade){
    srand(time(NULL));
    int sorrindo,piscando, dormindo, surpresa;

    quantidade = verificaQuantidade(quantidade);

    sorrindo = (rand()%quantidade);
    sorrindo = verificaQuantidade2(sorrindo);
    piscando = rand()%(quantidade-sorrindo);
    piscando = verificaQuantidade2(piscando);
    dormindo = rand()%(quantidade-piscando);
    dormindo = verificaQuantidade2(dormindo);
    surpresa = quantidade - (piscando + sorrindo + dormindo);
    surpresa = verificaQuantidade2(surpresa);
    printf("Sorrindo:%d \n Piscando:%d \n Dormindo:%d \n Surpresa:%d \n",sorrindo,piscando, dormindo,surpresa);
    criaObraSorrindo(matriz,sorrindo);
    criaObraPiscando(matriz,piscando);
    criaObraDormindo(matriz,dormindo);
    criaObraSurpresa(matriz,surpresa);
}
```

Figura 05. Função principal do gerador da obra de arte do aluno.

3 Resultados

Nesta seção iremos mostrar algumas capturas de tela da execução do programa. Para executar, além de ter um compilador de c, deve-se digitar no terminal dentro da pasta do projeto:

make all

Ou pode digitar o comando:

gcc main.c -o EXEC src/funcoes.c src/menu.c

E logo em seguida digitar o comando:

./EXEC

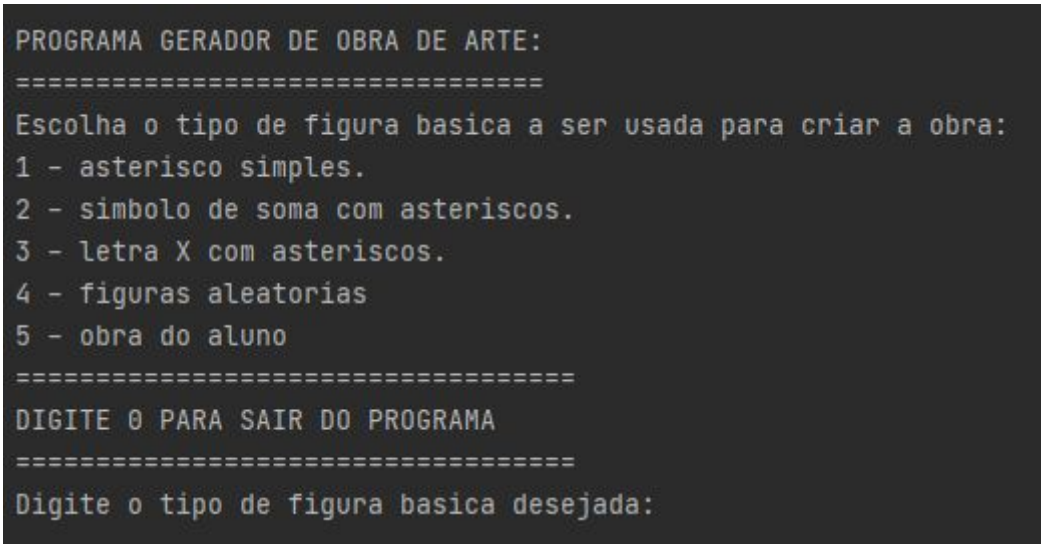


Figura 06. Menu do programa.

Abaixo temos a saída da opção 01 do menu, note que como o valor digitado para a quantidade foi menor que zero, um novo valor foi gerado. Observe que logo após quando o quadro é gerado, há a opção de criar um novo quadro com essas mesmas configurações.

Figura 07. Saída para a opção 01.

A seguir temos a saída para a opção 02 do programa, que novamente testamos para valores aleatórios:

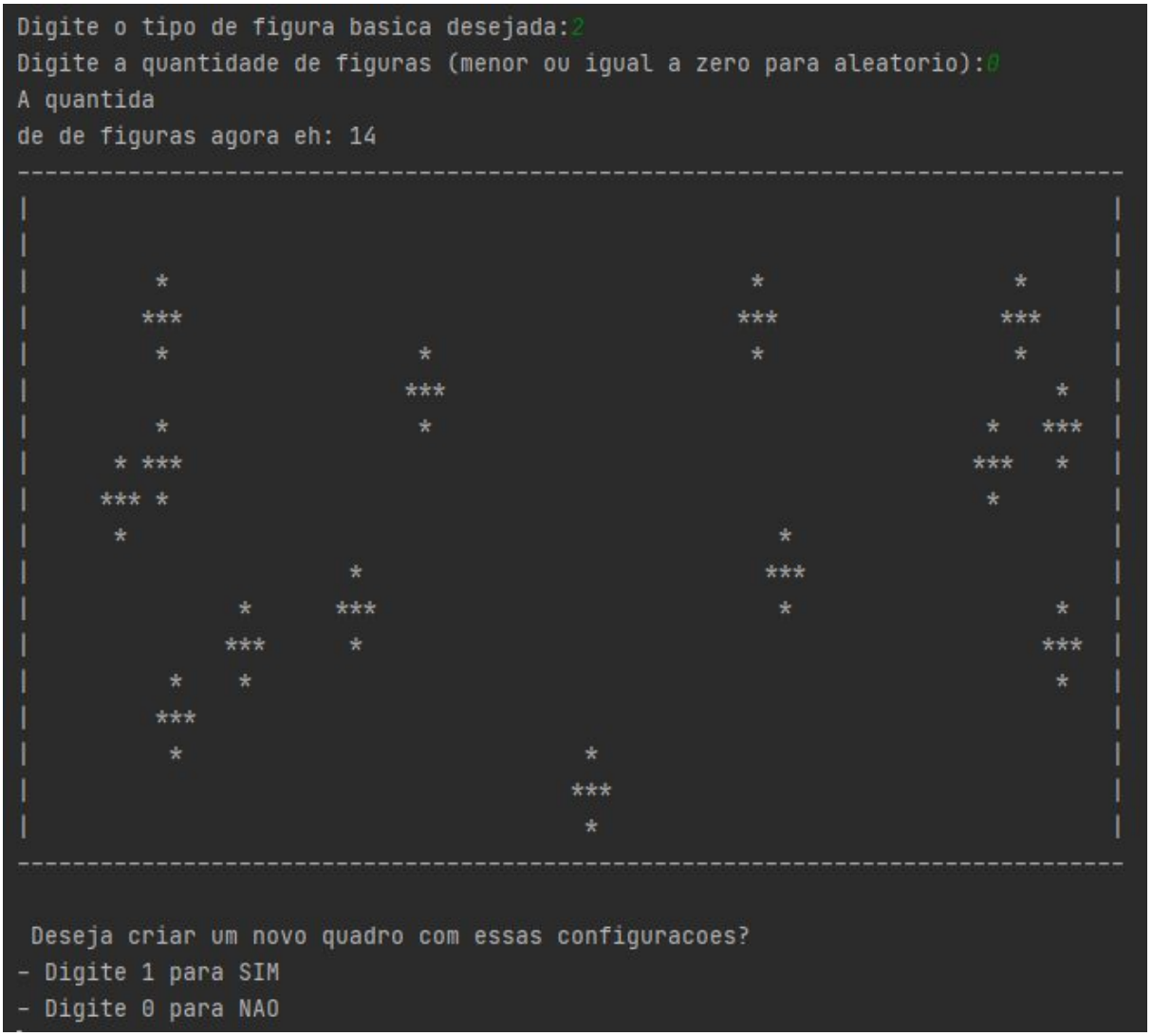


Figura 08. Saída para a opção 02.

Abaixo temos a saída para a opção 03, onde foi digitado uma quantidade de figuras maior que 100:

Figura 09. Saída para a opção 03.

Na figura abaixo temos a saída para a opção 04, que seria a geração de um quadro com todas as figuras das opções anteriores:

Figura 10. Saída para a opção 04.

Por último, a saída para a opção 05, criada pelo aluno:

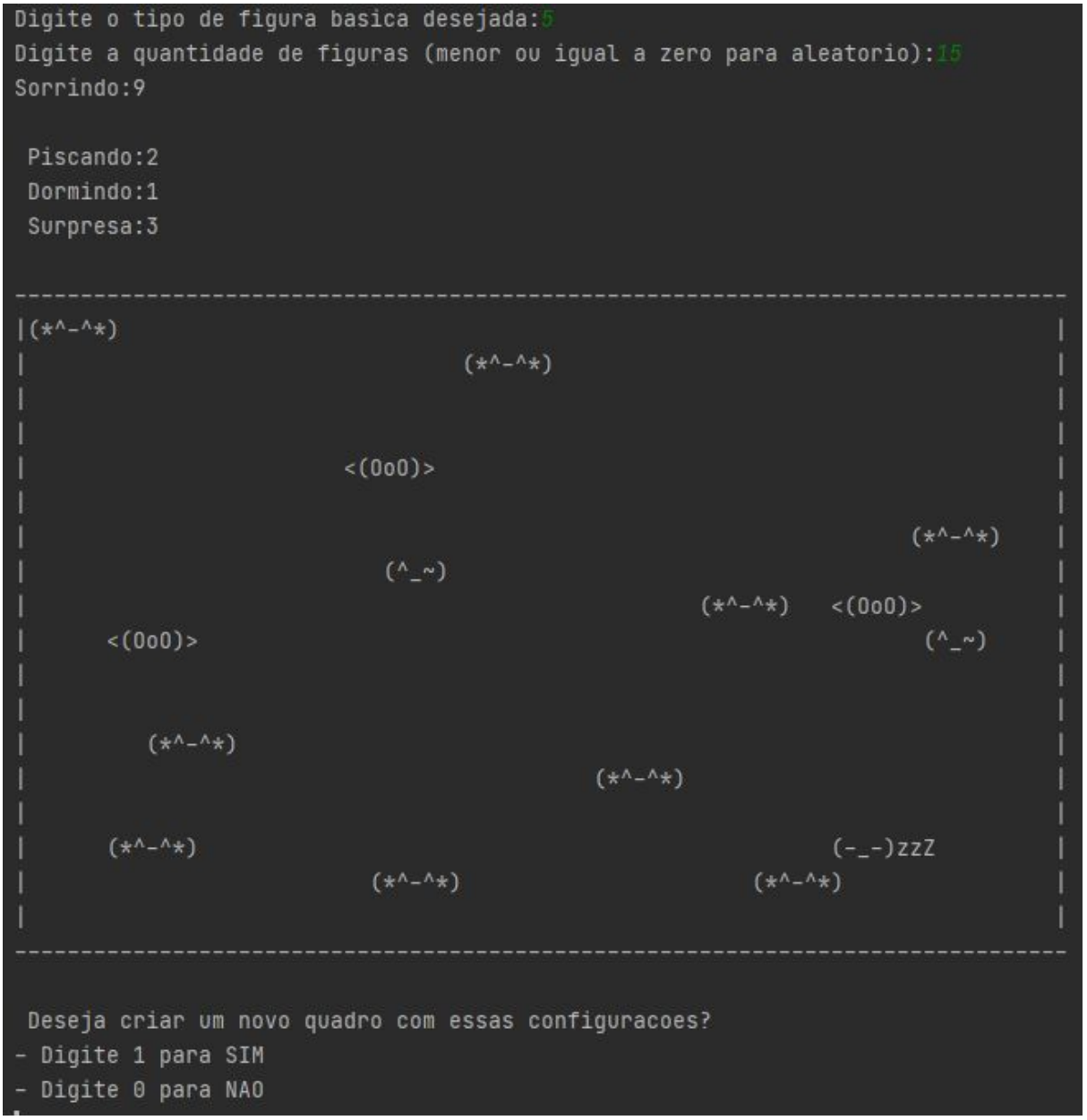


Figura 11. Saída para a opção 05.

4 Conclusão

O trabalho prático foi de extrema importância, para nós alunos termos um aquecimento com a linguagem C, já que o trabalho é uma boa forma de relembrarmos a linguagem e também exercitarmos o raciocínio lógico. Acredito que consegui cumprir todos os requisitos da especificação do trabalho prático.

No início houve algumas dificuldades sobre a lógica inicial, mas após alguns rascunhos e testes acabou dando certo e também houve dúvidas quanto a criação da obra do aluno, visto que eu gostaria de fazer algo divertido e gostei do resultado.

5 Referências

CASAVELLA, Eduardo. Valores aleatórios em C com a função rand. Disponível em: http://linguagemc.com.br/valores-aleatorios-em-c-com-a-funcao-rand/. Acesso em: 15 fev. 2021.

KAWANAMI, Silvia. Kaomoji, os emoticons japoneses. 2013. Disponível em: https://www.japaoemfoco.com/kaomoji-os-emoticons-japoneses/. Acesso em: 15 fev. 2021.

STEIN, Thaís. Kaomoji: os emojis japoneses. Disponível em: https://www.dicionariopopular.com/kaomoji-emojis-carinhas-kawaii/. Acesso em: 15 fev. 2021.