Universidade Federal de Viçosa - CAF

Projeto e Análise de Algoritmos - Ciência da Computação

André Elias Andrade Silva

Matrícula 3013

Trabalho Prático 0: Gerador aleatório de obras de arte

Florestal

2019

SUMÁRIO

LISTA DE IMAGENS

1. Introdução

Este trabalho consiste na implementação de um algoritmo que gera obras de arte aleatórias, sendo 5 tipos diferentes de arte, uma com apenas asteriscos (*), com um símbolo de mais (+) formado com asteriscos, um X com asteriscos, uma mistura dos 3 e, por fim, figuras escolhidas e feitas por mim, que foram rostos diferentes gerados com caracteres, dando assim a impressão de uma multidão ou grupo de pessoas, dependendo da quantidade de pessoas inseridas.

O que foi feito para criação do algoritmo foi a criação do menu de opções iniciais, dando ao usuário a oportunidade de escolher a opção de quadro desejada. Logo em seguida a criação do algoritmo em si, foi utilizada a implementação de uma TAD com as funções necessárias, chamada de “funcoes.h” e “funcoes.c”, fazendo assim as devidas abstrações do programa.

2. Desenvolvimento

O algoritmo começa, após ler a opção do usuário, criando um quadro inicial vazio. Esse quadro é feito utilizando uma matriz de 20 linhas por 80 colunas, através de comparações é armazenado o devido valor nas várias posições de matriz, se a posição comparada é uma borda da matriz é armazenado o caractere “-” se for a primeira ou última linha da matriz, e “|” se for a primeira e última coluna, com exceção da primeira posição de linha da coluna, pois, a mesma já foi armazenada com o caractere “-” anteriormente. As demais posições são armazenada o caractere vazio, o espaço, “ “. Isso tudo é feito por um método chamado “criaQuadro”.

```
void criaQuadro(char quadro[20][80]){
    int i,j;
    for (i=0;i<20;i++){
            for(j=0;j<80;j++){
                if (j == 0 || j == 79)
                    quadro[i][j] = '|';
                if (i == 0 || i == 19)
                    quadro[i][j] = '-';
                if (i != 0 && i!=19 && j!= 0 && j!= 79)
                    quadro [i][j] = ' ';
            }
    }

}
```

Imagem 1 – Cria quadro

A opção do usuário, após lida, é comparada com um switch para decidir qual obra de arte será feita. Então baseado nessa opção é chamada as funções referidas devidamente. Exemplo da primeira opção:

```
switch (opcao){
    case 1:
        criaQuadro(quadro);
        fazFigural(quadro, qtd);
        imprimeQuadro(quadro);
        refazQuadro(quadro, qtd, opcao);
    break;
```

Imagem 2 – Switch inicial

A função fazFigura1 funciona da seguinte maneira, o quadro é passado como parâmetro, como também, a quantidade de figuras desejadas pelo usuário. Então é feito um while que será usado para preencher as lacunas vazias, que vão ser geradas aleatoriamente pela função rand, onde pode ser inserida a imagem desejada, no caso, um asterisco. E o while só será finalizado quando todas as figuras foram devidamente alocadas no quadro, por isso, a quantidade só é decrementada quando for de fato armazenado um caractere, evitando assim que posições já armazenadas sejam sobrescrevidas.

```
void fazFigural(char quadro[20][80], int quantidade){
    while (quantidade > 0){
        int posicaoRand1 = rand()%20;
        int posicaoRand2 = rand()%80;
        if (quadro[posicaoRand1][posicaoRand2] == ' '){
            quadro[posicaoRand1][posicaoRand2] = '*';
            quantidade--;
        }
    }
}
```

Imagem 3 – fazFigura1

O método fazFunção2 e as demais seguem a mesma lógica, apenas com mais comparações, onde se compara a posição gerada pelo rand e verifica se os vizinhos referentes ao desenho são válidos, se possui um lugar livre para colocar o desenho.

```
void fazFigura2(char quadro[20][80], int quantidade){
    while (quantidade > 0){
        int posicaoRand1 = rand()%20;
        int posicaoRand2 = rand()%80;
        if (quadro[posicaoRand1][posicaoRand2] == ' ' && quadro[posicaoRand1-1][posicaoRand2] == ' ' &&
            quadro[posicaoRand1+1][posicaoRand2] == ' ' && quadro[posicaoRand1][posicaoRand2-1] == ' ' &&
            quadro[posicaoRand1][posicaoRand2+1] == ' '){

            quadro[posicaoRand1][posicaoRand2] = '*';
            quadro[posicaoRand1-1][posicaoRand2] = '*';
            quadro[posicaoRand1+1][posicaoRand2] = '*';
            quadro[posicaoRand1][posicaoRand2-1] = '*';
            quadro[posicaoRand1][posicaoRand2+1] = '*';
            quantidade--;
        }
    }
}
```

Imagem 4 - fazFigura2

A diferença entre os métodos de se fazer a obra aleatória aparece no método de se fazer a obra com as 3 figurar possíveis, onde é gerado um número aleatório entre 0 a 2, sendo essas as 3 figuras possíveis, o valor gerado entra em um switch onde será devidamente feito a alocação da imagem sorteada, sendo feito através da sua função, destacando que a quantidade passada como parâmetro é 1, pois só uma imagem vai ser alocada, as outras vão ser alocadas enquanto a quantidade da função fazRandom existir.

```c
void fazRandom(char quadro[20][80], int quantidade){
    int figura;
    while (quantidade > 0){
        figura = rand() % 3;
        switch(figura){
            case 0:
                fazFigura1(quadro, 1);
            break;
            case 1:
                fazFigura2(quadro, 1);
            break;
            case 2:
                fazFigura3(quadro, 1);
            break;
        }
        quantidade--;
    }

}
```

Imagem 5 - fazRandom

A função de ser fazer a figura de escolha do aluno foi feito seguindo a mesma lógica mostrada anteriormente, onde se utiliza o switch entre 3 números aleatórios para se determinar a imagem desejada, e alocando a imagem comparando a respectiva posição também gerada aleatoriamente e as devidas posições adjacentes.

Exemplo de execução:

```
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 especial
5
Digite a quantidade de figuras:
10
--------------------------------------------------------------------------------
|                                                                              |
|                                                                              |
|                                                                              |
|    /(0_0)/              |(>.<)/                                              |
|                                                                              |
|                                                                              |
|                                                  |(^_^)|                     |
|                                                                              |
|                                                             /(0_0)/          |
|                          |(>.<)/                                             |
|                             /(0_0)/                                          |
|                                                  /(0_0)/                     |
|                 /(0_0)/                                                      |
|                                                                              |
|        /(0_0)/                                                               |
|                                                                              |
|                       |(^_^)|                                                |
|                                                                              |
--------------------------------------------------------------------------------
Deseja refazer o quadro? Y/N
```

Imagem 6 – Teste de execução

Após a criação do quadro é perguntado ao usuário se ele deseja refazer o quadro, se sim, o quadro é refeito a partir do zero, onde todas os caracteres de borda são alocados e os respectivos espaços vazios (espaços).

E isso é feito recursivamente enquanto o usuário digitar não. Para isso foi feito uma função para refazer o quadro, e ela foi criada da seguinte maneira:

```
void refazQuadro(char quadro[20][80], int quantidade, int opcao){
    char refazer;
    printf("Deseja refazer o quadro? Y/N\n");
    fflush(stdin);
    scanf("%c", &refazer);
    if (refazer == 'Y' || refazer == 'y'){
        switch(opcao){
            case 1:
            criaQuadro(quadro);
            fazFigural(quadro, quantidade);
            imprimeQuadro(quadro);
        break;
        case 2:
            criaQuadro(quadro);
            fazFigura2(quadro, quantidade);
            imprimeQuadro(quadro);
        break;
```

Imagem 7 - refazQuadro

O switch contém 5 opções diferentes, onde cada um representa a respectiva opção de obra de arte, que foram abstraídas para a imagem ficar mais compacta.

```
        case 5:
            criaQuadro(quadro);
            fazEspecial(quadro, quantidade);
            imprimeQuadro(quadro);
        break;
        }
        refazQuadro(quadro, quantidade ,opcao);
    } else if (refazer == 'N' || refazer == 'n'){
        return;
    } else {
        printf("Opcao invalida\n");
        refazQuadro(quadro, quantidade, opcao);
    }
}
```

Imagem 8 – final do refazQuadro

E termina chamando a função recursivamente enquanto o usuário digitar N (Não).

3. Conclusão

No geral, este trabalho foi muito produtivo, onde consegui desenvolver o que foi pedido com agilidade. Achei interessante o algoritmo em si onde, até então, não havia desenvolvido nada parecido, de utilizar acontecimentos aleatórios e como tratar os conflitos.

No fim, foi feito tudo o que foi pedido pela descrição do trabalho, onde tudo funciona avaliando os resultados de execução do algoritmo.

4. Referências

Citethisform. **Guia Prático de Citações e Referências Segundo as Normas ABNT**. Belo Horizonte, MG. Disponível em: <http://www.citethisforme.com/pt/normas-abnt>, Acesso em: 24 ago. 2019.

FASTFORMAT. **Como referenciar figuras e tabelas nas normas ABNT?**. Belo Horizonte, MG. Disponível em: <https://blog.fastformat.co/como-referenciar-figuras-tabelas-normas-abnt/>, Acesso em: 24 ago. 2019.