UNIVERSIDADE FEDERAL DE VIÇOSA — CAMPUS FLORESTAL
Ciência da Computação Projeto e
Análise de Algoritmos

Pablo Ferreira

03480

TRABALHO PRÁTICO 0

Florestal 2019

UNIVERSIDADE FEDERAL DE VIÇOSA — CAMPUS FLORESTAL

Pablo Ferreira

03480

TRABALHO PRÁTICO 0

Relatório prático apresentado a disciplina de projeto e análise de algoritmos, do curso Ciência da Computação da Universidade Federal de Viçosa — Campus Florestal.

Prof.: Daniel Mendes Barbosa

Florestal 2019

SUMÁRIO

1 INTRODUÇÃO

O problema proposto tinha a seguinte objetivo, o desenvolvimento de um algoritmo que usasse números aleatórios para gerar um quadro com artes. Esse quadro deveria ter 3 artes básicas formadas por asteriscos, 1 arte gerada aleatoriamente com as figuras 1,2 e 3, e 1 arte feita pelo aluno. As figuras básicas são, um asterisco simples, um símbolo do sinal de soma, e a letra X.

O tamanho de entrada do programa deve ser um número de 1 a 100, e as artes devem ser posicionadas sempre de maneira aleatória dentro do quadro, de forma que quando o usuário digitasse um número menor que 1, o programa geraria um novo número aleatório entre 1 e 100 e quando fosse digitado um número maior que 100, o programa reconheceria como 100.

A opção 5 segue os mesmos conceitos de implementação das opções anteriores, tendo como diferença o seu formato, que é decidido pelo desenvolvedor(aluno) do programa.

2 DESENVOLVIMENTO

Para gerar as posições aleatórias, foi necessário duas funções, a função *srand((unsigned)time(NULL))* e a função *rand(),* e para declarar o intervalo dos números aleatórios, sempre utilizamos uma variável para receber o "*mod"* da função rand() pelo número que desejamos que seja o limite, exemplo, números de 0 a 10 podem ser recebidos com o seguinte código:

aux = rand() % 10;

Após entender o funcionamento de qualquer uma das funções de inserção, as outras funções de inserção foram implementadas seguindo a mesma lógica, apenas alterando os condicionais.

Para uma melhor organização do programa, foram criados 2 arquivos ".h". Sendo um para as funções do quadro, e um para as funções do menu.

Os arquivos "funções.c" e "funções.h" contém as seguintes funções:

Figura 1

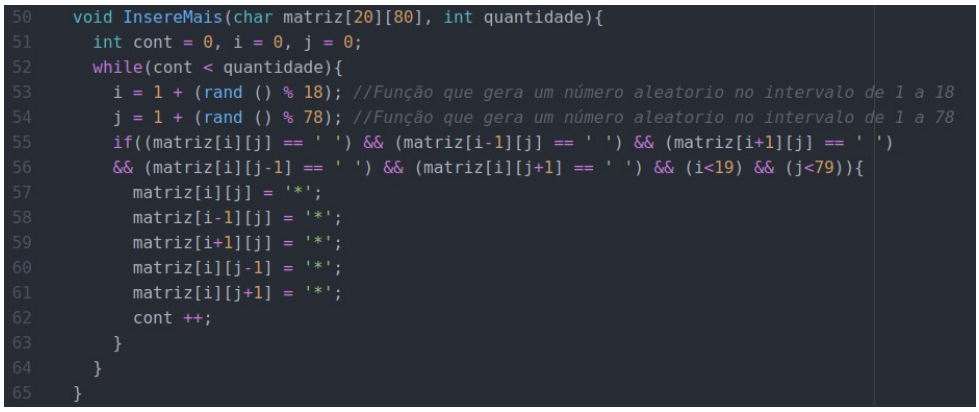
```
#ifndef FUNCOES_H
#define FUNCOES_H

//Prototipos das funções

void IniciarQuadro(char matriz[20][80]);
void InsereAsterisco(char matriz[20][80], int quantidade);
void InsereMais(char matriz[20][80], int quantidade);
void InsereX(char matriz[20][80], int quantidade);
void InserirAleatorio(char matriz[20][80], int quantidade);
void InserirObraAluno(char matriz[20][80], int quantidade);
void ConfereQuantidade(int *quantidade);
void ConfereQuantidadePacman(int *quantidade);
void ImprimirQuadro(char matriz[20][80]);


#endif
```

Fonte: funcao.h

Onde a função *IniciarQuadro* inicia uma matriz[20][80] com espaços vazios e preenche as bordas verticais com "|" e as horizontais com "-", representando assim, um quadro.

As funções *InsereAsterisco*, *InsereMais* e *InsereX,* inserem, respectivamente, as artes de um asterisco, um símbolo do sinal de soma e uma letra X. A criação de ambas são bem parecidas, apenas diferenciando os condicionais.

Figura 2



```
50  void InsereMais(char matriz[20][80], int quantidade){
51    int cont = 0, i = 0, j = 0;
52    while(cont < quantidade){
53      i = 1 + (rand () % 18); //Função que gera um número aleatorio no intervalo de 1 a 18
54      j = 1 + (rand () % 78); //Função que gera um número aleatorio no intervalo de 1 a 78
55      if((matriz[i][j] == ' ') && (matriz[i-1][j] == ' ') && (matriz[i+1][j] == ' ')
56      && (matriz[i][j-1] == ' ') && (matriz[i][j+1] == ' ') && (i<19) && (j<79)){
57        matriz[i][j] = '*';
58        matriz[i-1][j] = '*';
59        matriz[i+1][j] = '*';
60        matriz[i][j-1] = '*';
61        matriz[i][j+1] = '*';
62        cont ++;
63      }
64    }
65  }
```

Fonte: funcoes.c

A função *InserirAleatorio* divide a quantidade digitada pelo o usuário em 3 valores aleatórios que são passados como parâmetro nas funções anteriores.

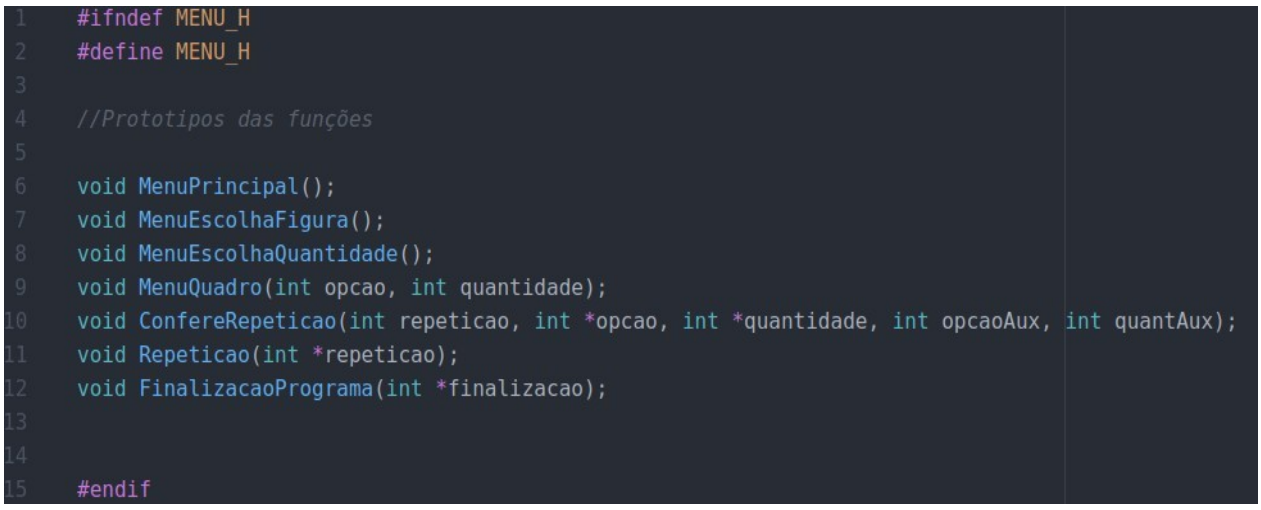
A função *InserirObraAluno* insere a obra criada pelo aluno, que no caso, é um desenho de um PacMan.

As funções *ConfereQuantidade* e *ConfereQuantidadePacman* conferem as regras de quantidade de artes limitadas pelo desenvolvedor. Como o desenho do PacMan é um pouco maior que os outros, a quantidades de artes que podem ser exibidas foi de 17, evitando loops infinitos por falta de espaço.

E por último, a função *ImprimirQuadro* imprime a matriz completa com as obras de artes geradas.

Os arquivos “menu.c” e “menu.h” contém as seguintes funções:

Figura 3



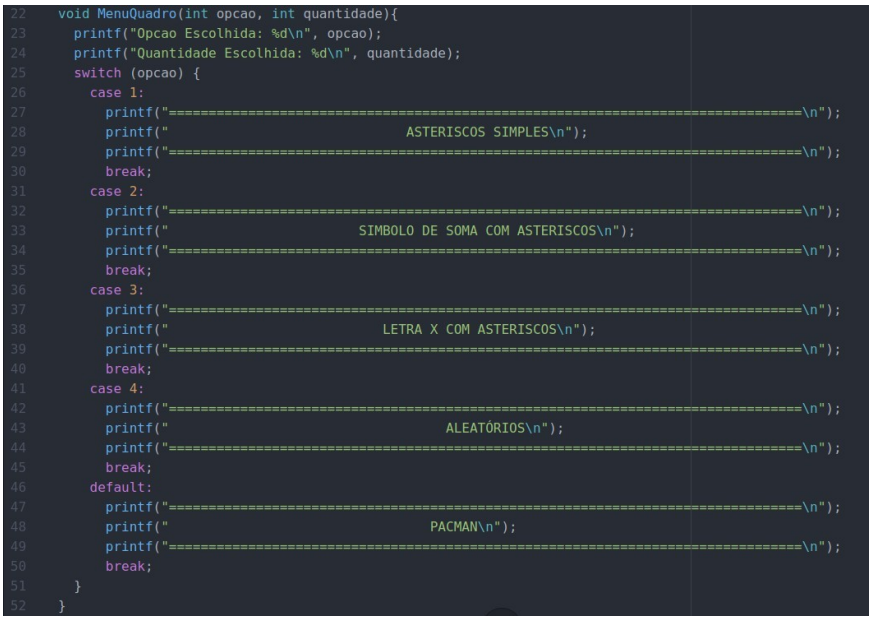
```
1   #ifndef MENU_H
2   #define MENU_H
3
4   //Prototipos das funções
5
6   void MenuPrincipal();
7   void MenuEscolhaFigura();
8   void MenuEscolhaQuantidade();
9   void MenuQuadro(int opcao, int quantidade);
10  void ConfereRepeticao(int repeticao, int *opcao, int *quantidade, int opcaoAux, int quantAux);
11  void Repeticao(int *repeticao);
12  void FinalizacaoPrograma(int *finalizacao);
13
14
15  #endif
```

Fonte: menu.h

Onde as funções *MenuPrincipal*, *MenuEscolhaFigura*, *MenuEscolhaQuantidade* imprimem, respectivamente, menus de opções que podem ser escolhidas pelo usuário, menu para o usuário digitar a figura desejada e menu para digitar a quantidade desejada.

A função *MenuQuadro* recebe como parâmetro a arte e a quantidade selecionada pelo usuário, e então imprime o nome do quadro e a quantidade selecionada.

Figura 4



```
void MenuQuadro(int opcao, int quantidade){
  printf("Opcao Escolhida: %d\n", opcao);
  printf("Quantidade Escolhida: %d\n", quantidade);
  switch (opcao) {
    case 1:
      printf("================================================================================\n");
      printf("                              ASTERISCOS SIMPLES\n");
      printf("================================================================================\n");
      break;
    case 2:
      printf("================================================================================\n");
      printf("                       SIMBOLO DE SOMA COM ASTERISCOS\n");
      printf("================================================================================\n");
      break;
    case 3:
      printf("================================================================================\n");
      printf("                           LETRA X COM ASTERISCOS\n");
      printf("================================================================================\n");
      break;
    case 4:
      printf("================================================================================\n");
      printf("                                 ALEATÓRIOS\n");
      printf("================================================================================\n");
      break;
    default:
      printf("================================================================================\n");
      printf("                                PACMAN\n");
      printf("================================================================================\n");
      break;
  }
}
```

Fonte: menu.c

A função *ConfereRepeticao* confere se o usuário selecionou que o programa repita o quadro feito anteriormente e então copia os valores usados para gerar esse novo quadro.

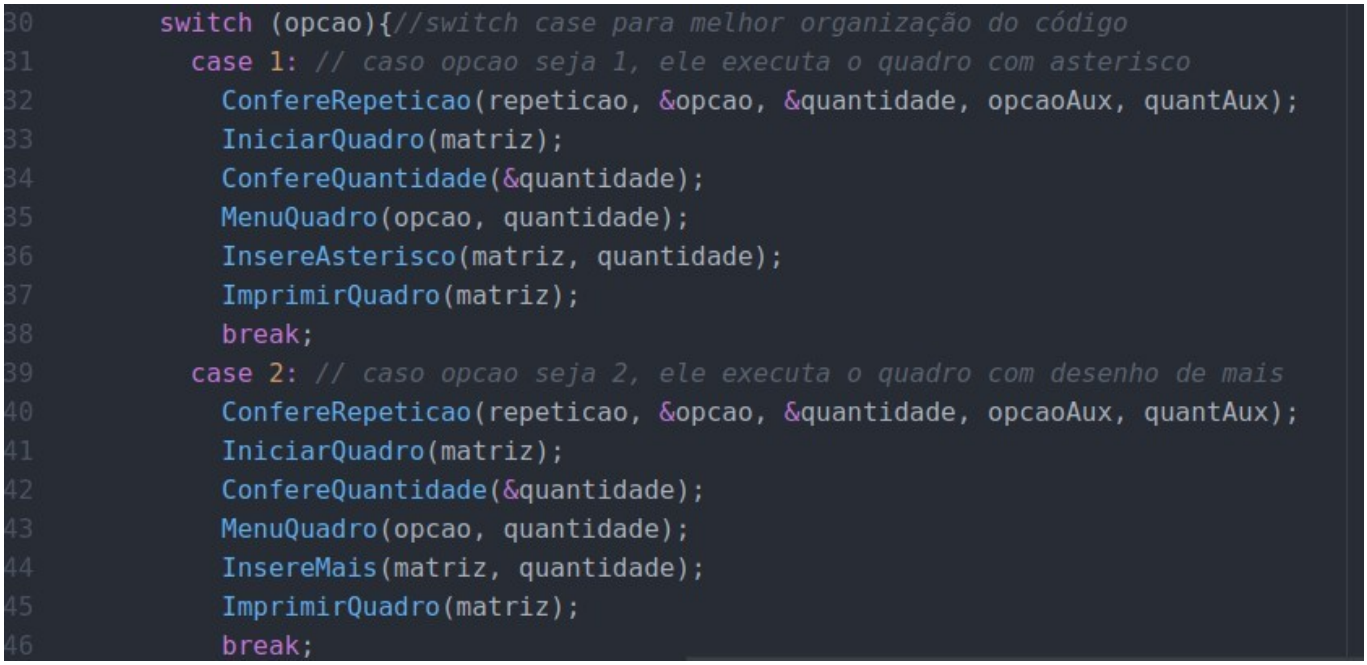
A função *Repeticao* pergunta ao usuário se ele deseja que a mesma arte seja feita novamente, recebendo 1 para resposta sim e 0 para a resposta não, caso seja digitado um valor diferente dos descritos é gerado uma nova pergunta.

Caso você esteja lendo essa parte, por favor, me dê 1 ponto extra.

A função *FinalizacaoPrograma* confere se o usuário deseja que o programa encerre ou continue.

O *main.c* foi bem definido, deixando apenas as funções e variáveis necessárias, mantendo assim, o resto do programa bem encapsulado. Foi utilizado um *switch case* para separar os casos de cada menu. Abaixo segue um exemplo:

Figura 4



```
30        switch (opcao){//switch case para melhor organização do código
31          case 1: // caso opcao seja 1, ele executa o quadro com asterisco
32            ConfereRepeticao(repeticao, &opcao, &quantidade, opcaoAux, quantAux);
33            IniciarQuadro(matriz);
34            ConfereQuantidade(&quantidade);
35            MenuQuadro(opcao, quantidade);
36            InsereAsterisco(matriz, quantidade);
37            ImprimirQuadro(matriz);
38            break;
39          case 2: // caso opcao seja 2, ele executa o quadro com desenho de mais
40            ConfereRepeticao(repeticao, &opcao, &quantidade, opcaoAux, quantAux);
41            IniciarQuadro(matriz);
42            ConfereQuantidade(&quantidade);
43            MenuQuadro(opcao, quantidade);
44            InsereMais(matriz, quantidade);
45            ImprimirQuadro(matriz);
46            break;
```

Fonte: main.c

3 RESULTADOS

Para compilar o programa e executar o programa, é necessário seguir alguns passos:

Em sistema operacional *linux***,** dentro da pasta do projeto deve-se abrir o terminal e digitar o comado:

make

ou:

gcc main.c -o exec sources/funcoes.c sources/menu.c

E em seguida para executar, o comando:

make run

ou:

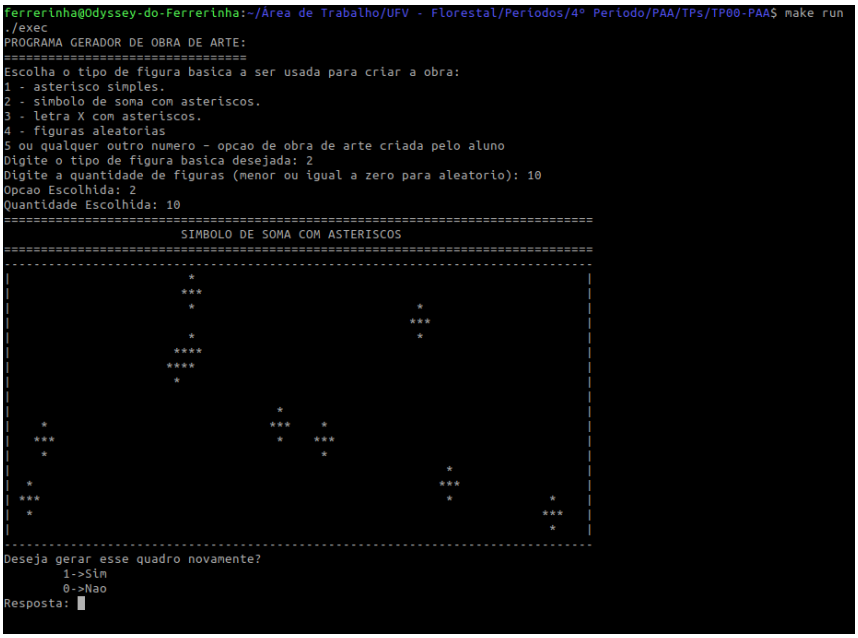
./exec

Segue abaixo alguns exemplos de quadros gerados pelo programa:

Figura 5

Fonte: Terminal Linux

Figura 6



Fonte: Terminal Linux

Figura 7

Fonte: Terminal Linux

Figura 8

Fonte: Terminal Linux

Figura 9

Fonte: Terminal Linux

4 DIFICULDADES

Uma das dificuldades foi encontrar a lógica inicial de como colocar os desenhos na matriz. Também houve erros de sintaxe e erros de lógica. Fora isso, o programa se desenvolveu bem, sendo implementado rapidamente, resultando no objetivo esperado.

Para criar o desenho do aluno, foi necessário fazer o desenho com caracteres em um bloco de notas e após isso, como era um desenho complicado, foi necessário desenhar uma matriz do tamanho de espaços necessários para se fazer o desenho, e então desenhar os caracteres nas posições desejadas, após isso, é selecionado uma posição da matriz para ser a posição central e então é calculado as posições seguintes até o desenho estar formado.

5 CONCLUSÃO

Aprendi sobre a função srand e o seu funcionamento, que através da hora do computador, ela gera um número aleatório, sendo possível, embora raro, que esse número se repita caso o intervalo de execução seja muito próximo. Sendo assim, todas as vezes, ao executar o programa, é gerado artes em posições diferentes. Com isso, mesmo que fossem selecionados a mesma arte e a mesma quantidade, sempre eram gerados quadros com artes em posições diferentes.

REFERÊNCIA

MOREIRA, Jarlisson. Gerando números aleatórios em C: rand, srand e seed, 2013. Disponível em: https://www.cprogressivo.net/2013/03/Como-gerar-numeros-aleatorios-em-C-com-a-rand-srand-e-seed.html. Acesso em: 12 de Agosto de 2019.