

**UNIVERSIDADE FEDERAL DE VIÇOSA - CAMPUS FLORESTAL**  
**CIÊNCIAS EXATAS**  
**DEPARTAMENTO DE COMPUTAÇÃO**

**Lucas Gabriel Barbosa Cunha**

**Trabalho 0**

**Florestal**  
**2019**

**Lucas Gabriel Barbosa Cunha**

**Trabalho 0**

**Florestal  
2019**

## Sumário

<b>1</b>	<b>Introdução . . . . .</b>	<b>3</b>
<b>2</b>	<b>Desenvolvimento . . . . .</b>	<b>4</b>
<b>3</b>	<b>Conclusão . . . . .</b>	<b>8</b>

## 1 Introdução

Devido ao periodo de ferias e o tempo passado longe ao computador, com o proposito de voltarmos a ativa e realizarmos um aquecimento, foi nos apresentado o seguinte trabalho.

O problema proposto foi o de fazer um programa, o qual gerasse um quadro aleatório com base na imagem escolhida pelo usuário, sendo ela(s):

- 1 - Asteriscos simples.
- 2 - Conjunto de asteriscos formando o simbolo de soma (+).
- 3 - Conjunto de asteriscos formando a letra X.
- 4 - Arte criativa.

No decorrer da documentação trataremos da abordagem utilizada para a execução de todos esses pontos.

## 2 Desenvolvimento

Para o desenvolvimento deste trabalho, fora utilizado o editor de texto “JetBrains CLion 2019.1.2”, juntamente com o compilador “cygwin64 gcc” e inicialmente como especificado pelo roteiro foi realizado a construção do quadro em branco.

Através de uma matriz com tamanhos 20x80 fiz, inicialmente, a moldura do quadro, com a utilização do seguinte trecho de código:

**Figura 1 – moldura do quadro**

```
//=====F. Quadro=====
for (i = 0; i < 20; i++) {
    for (j = 0; j < 80; j++) {
        if (i == 0 || i == 19) {
            matriz[i][j] = '-';
        }
        if (j == 0 || j == 79) { //Loop responsável por adaptar as posições da matriz
            matriz[i][j] = '|'; // ao formato do quadro
        }
        matriz[i][j] = '\n';
    }
}
//=====F. Quadro=====
```

O qual checa se a posição atual da matriz é uma borda inferior ou superior e a preenche com os caracteres “-” e “|” respectivamente.

Em seguida, foi realizado a construção do menu de seleção, onde o usuário é capaz de escolher o tipo de figura, juntamente com a quantidade de figuras que irão compor o quadro.

Logo após começou a abordagem de cada imagem que seria gerada. O quadro que seria feito apenas de simples asteriscos foi simples, o único cuidado necessário foi para que a posição gerada não possuísse um asterisco ou estivesse na borda do quadro, o problema foi resolvido com uma checagem onde era conferido se a posição gerada já estava ocupada.

**Figura 2 – Função responsável por gerar asteriscos no quadro**

```
//===== Gerar um asterisco simples =====
void fig0(char matriz[80], int p){
    int pos = 0;
    while (pos / 80 == 0 || pos / 80 == 19 || pos % 80 == 79 || pos % 80 == 0 || matriz[pos/80][pos%80] == '*') {
        pos = rand() % 1600;
    }
    matriz[pos / 80][pos % 80] = '*';
}
//===== Gerar um asterisco simples =====
```

A condição de parada do while é que a posição gerada não possua um asterisco

e nem se encontre na borda do quadro, e essa condição se tornou a condição generica para todas as outras imagens.

Para as imagens formadas por não apenas um asterisco mas sim um conjunto de asteriscos a abordagem foi a seguinte:

- 1) gerar uma posição inicial a qual seria a posição central da imagem.
- 2) colocar as condições genericas de parada do *while*.
- 3) adicionar condições com base no tamanho da imagem, ou seja além de verificar se a posição gerada esta vaga, foi verificado também se todas as casas as quais a figura utilizaria estavam vagas juntamente com que os arredores da posição gerada coubessem a figura a ser inserida, ou seja que a posição não estivesse muito proximo a bordas ou a outras figuras que pudessem comprometer alguma parte da imagem.

Abaixo se encontra o *while* responsavel por gerar a posição a qual seria o ponto central da letra X:

**Figura 3 – emphWhile gerador da posição central da figura X**

```
//===== Gerar a letra X =====  
void fig2(char matriz[][80], int n) {  
    int pos = 0;  
    while (pos / 80 == 0 || pos / 80 == 19 || pos % 80 == 79 || pos % 80 == 0 || (pos / 80 - 1) == 0 ||  
           (pos / 80 + 1) == 19 || (pos % 80 + 1) == 79 || (pos % 80 - 1) == 0 ||  
           matriz[pos / 80][pos % 80] == '*' || matriz[pos / 80 - 1][pos % 80 - 1] == '*' ||  
           matriz[pos / 80 + 1][pos % 80 + 1] == '*' || matriz[pos / 80 - 1][pos % 80 + 1] == '*' ||  
           matriz[pos / 80 + 1][pos % 80 - 1] == '*') {  
        pos = rand() % 1600;  
    }  
}
```

Como pode-se notar, quanto mais complexa for a imagem, mais verificações será necessario realizar, afim de garantir que todas as posições ocupadas pela imagem estejam livres.

A partir daí a unica preocupação foi com que as condições estivessem corretas, afim de não prejudicar a integridade de outras imagens geradas, essa abordagem foi utilizada para gerar todas as imagens.

Para a quinta opção de imagem, foi escolhido o tema Space Invaders, um jogo presente em celulares antigos da NOKIA.

a abordagem foi a mesma, porém além de ser possivel escolher o numero de naves presentes no quadro o usuário pode optar por escolher o número de asteroides presentes na imagem.







### **3 Conclusão**

Esse trabalho foi um excelente exercício para que nos, alunos, voltássemos a “ativa” afim de exercitarmos nosso cérebro, foi uma atividade interessante uma vez que os focos dos “TPs” são normalmente voltados para uma área mais abstrata, então foi divertido e interessante ver algo sendo construído ilustrativamente.