UNIVERSIDADE FEDERAL DE VIÇOSA - CAMPUS FLORESTAL

Trabalho prático de Projeto e Análise de Algoritmos

Nome: Luana Tavares Anselmo
Matrícula: 5364

Florestal - MG
2024

UFV

Sumário

1. Introdução **3**

2. Metodologia **4**

2.1. Uso de números aleatórios

2.2. Escolha do desenho

3. Resultados do desenvolvimento **6**

3.1. Problemas de implementação

4. Compilação **8**

5. Considerações finais **9**

6. Referências bibliográficas **10**

UFV

1. Introdução

O Trabalho Prático 0 da disciplina de Projeto e Análise de Algoritmos tem como objetivo explorar conceitos fundamentais de programação em C, combinados com elementos de arte digital. O projeto envolve o desenvolvimento de um programa que gera "obras de arte" aleatórias, utilizando uma figura geométrica simples, o asterisco, como base para criar padrões visuais.

O objetivo principal do trabalho é proporcionar ao usuário uma experiência interativa, onde ele pode escolher a quantidade e o tipo de figuras que serão geradas, além de explorar a variabilidade das composições. Graças à implementação de números aleatórios no programa, cada execução resulta em um quadro único, com as figuras distribuídas em posições diferentes, garantindo que o resultado seja sempre imprevisível.

O projeto envolve a aplicação de algoritmos que manipulam matrizes em um contexto prático, onde cada tipo de desenho possui um padrão específico, assim, envolvendo cálculos de coordenadas. O desafio de distribuir as figuras sem sobreposição em um espaço limitado permite o uso de estratégias no algoritmo para otimizar a ocupação da matriz.

2. Metodologia

2.1. Uso de números aleatórios

No **Trabalho Prático 0** da disciplina de **Projeto e Análise de Algoritmos**, os números aleatórios desempenham um papel essencial na geração das obras de arte. Eles foram utilizados para criar variações nas composições visuais a cada execução do programa, garantindo que as figuras geométricas, como o asterisco, apareçam em posições diferentes a cada vez.

- **Posicionamento Aleatório das Figuras:** O uso da função rand( ) permitiu que o programa gerasse coordenadas `x` e `y` aleatórias dentro de uma matriz que representa o quadro onde as figuras são desenhadas. A cada execução do programa, o número gerado aleatoriamente define em qual ponto da matriz a figura será posicionada. Isso garante que, mesmo que o usuário escolha o mesmo tipo e quantidade de figuras, o resultado final será diferente, pois as figuras ocuparão posições distintas em cada execução.

- **Quantidade de Figuras Aleatória:** Além do posicionamento, os números aleatórios também foram usados para determinar a quantidade de figuras que serão desenhadas. Caso o usuário opte por não especificar uma quantidade fixa, o programa gera um número aleatório dentro de um intervalo, nesse caso entre 1 e 100.

- **Tipo de Figura Aleatória:** Quando o usuário escolhe a opção de figuras aleatórias, a função rand () também é utilizada para selecionar aleatoriamente entre diferentes tipos de figuras pré-definidas. Dessa forma, o programa pode alternar entre símbolos como asteriscos, somas,letras "X" ou casas, criando um padrão artístico diversificado.

2.2 Escolha do desenho

A escolha de implementar uma casa como uma das figuras no Trabalho Prático 0 surgiu da necessidade de criar um padrão mais elaborado e visualmente interessante, além dos formatos geométricos simples como o asterisco e a letra "X". A figura da casa foi selecionada por ser um símbolo facilmente reconhecível.

A casa foi desenhada utilizando asteriscos para representar as principais partes de uma construção: base, paredes, porta e telhado. A ideia era criar uma figura mais complexa, composta por vários elementos que ocupam um bloco maior de posições na matriz do quadro.

- **Base da Casa:** A base foi desenhada com 7 asteriscos em linha, formando a parte inferior da construção.

- **Paredes e Porta:** As paredes da casa foram implementadas em torno da porta central, com dois espaços em branco para representar a entrada, enquanto as laterais contêm asteriscos para formar as colunas que sustentam a estrutura. Isso trouxe um elemento mais dinâmico ao desenho, além de um desafio para garantir que os asteriscos fossem corretamente posicionados e que a porta central fosse exibida.

- **Telhado:** O telhado da casa foi desenhado em um formato triangular, com 5 asteriscos formando a parte superior e um único asterisco no topo representando a ponta.

O principal desafio ao desenhar a casa foi garantir que ela fosse corretamente representada em uma matriz bidimensional, respeitando o espaço ocupado pela figura. Diferente das outras figuras simples, a casa exige um espaço maior e mais estruturado, o que requer uma verificação mais detalhada das coordenadas antes de desenhá-la, para evitar que elementos da figura sejam cortados ou posicionados fora da matriz.

Outro aspecto importante foi a escolha das proporções. A casa precisava ser suficientemente grande para ser reconhecível e esteticamente equilibrada, mas não tão grande a ponto de ocupar todo o quadro. O desenho foi ajustado para garantir que ele pudesse ser inserido em diferentes posições dentro da matriz, mantendo sua integridade visual.

Exemplo de saída: Desenho de 25 casas

3. Resultados do desenvolvimento

3.1. Problemas de implementação

Durante o desenvolvimento do projeto, enfrentei alguns problemas relacionados ao gerenciamento de espaço no quadro e à sobreposição de figuras. Esses desafios surgiram devido à necessidade de posicionar várias figuras geométricas em uma matriz limitada, sem que elas se sobrepusessem, o que tornaria a "obra de arte" ilegível e confusa. A solução desses problemas exigiu o desenvolvimento de funções específicas para verificar e controlar o uso de espaço no quadro, garantindo que cada figura fosse desenhada corretamente e sem interferir nas outras.

1. **Espaço Limitado na Matriz:** Como o quadro onde as figuras são desenhadas é representado por uma matriz bidimensional, havia uma limitação de espaço disponível para inserir as figuras. À medida que mais figuras eram geradas, o risco de não haver mais espaço livre aumentava. Além disso, algumas figuras, como a casa, ocupam um espaço consideravelmente maior que as outras, o que exigiu um controle ainda mais rigoroso para garantir que toda a figura pudesse ser desenhada sem sair dos limites da matriz.

2. **Sobreposição de Figuras:** Outro problema comum foi a sobreposição de figuras. Como as coordenadas das figuras eram geradas aleatoriamente, existia a possibilidade de que uma nova figura fosse desenhada sobre uma já existente. Isso não só afetava a estética da obra de arte como também tornava o resultado incoerente. O desafio era garantir que as novas figuras fossem colocadas apenas em espaços livres do quadro.

Soluções Implementadas:

Para resolver esses problemas, foram desenvolvidas funções que verificam o espaço disponível no quadro antes de desenhar uma nova figura. Essas funções foram essenciais para garantir que as figuras fossem posicionadas de forma adequada, respeitando os limites da matriz e evitando sobreposições.

- ***Função areaDisponivel***

  Função `areaDisponivel`:A função `areaDisponivel()` foi criada para verificar se a área onde a nova figura seria desenhada estava livre. Ela recebe como parâmetros as coordenadas `x` e `y` da figura, além do tipo de figura que será desenhada. Dependendo do tipo, a função checa se todas as posições da matriz necessárias para a figura estão vazias

```c
int areaDisponivel(Quadro *quadro, int x, int y, int tipoFigura) {
    switch (tipoFigura) {
        case 1: // Asterisco simples
            return quadro->matriz[x][y] == ' ' &&
                   quadro->matriz[x][y - 1] == ' ' &&
                   quadro->matriz[x][y + 1] == ' ' &&
                   quadro->matriz[x - 1][y] == ' ' &&
                   quadro->matriz[x + 1][y] == ' ';

        case 2: // Símbolo de soma
            return quadro->matriz[x][y] == ' ' &&
                   quadro->matriz[x - 1][y] == ' ' &&
                   quadro->matriz[x + 1][y] == ' ' &&
                   quadro->matriz[x][y - 1] == ' ' &&
                   quadro->matriz[x][y + 1] == ' ' &&
                   quadro->matriz[x - 1][y - 1] == ' ' &&
                   quadro->matriz[x - 1][y + 1] == ' ' &&
                   quadro->matriz[x + 1][y - 1] == ' ' &&
                   quadro->matriz[x + 1][y + 1] == ' ';
```

Parte da função areaDisponivel que verifica se cada posição dos desenhos possuem espaço vazio

- ***Controle de Limites do Quadro:***

Para evitar que as figuras fossem desenhadas parcialmente fora do quadro, foi implementado um controle de limites nas funções de geração de coordenadas aleatórias. Esse controle assegura que as coordenadas geradas estejam dentro dos limites da matriz, especialmente para figuras que ocupam mais espaço, como a casa.

```c
do {
    x = rand() % (LINHAS - 2) + 2;
    y = rand() % (COLUNAS - 4) + 2;
```

As coordenadas de `x` e `y` para o centro de uma figura são ajustadas para que não fiquem próximas demais das bordas do quadro, permitindo que todas as partes da figura caibam dentro da matriz sem cortar nenhum pedaço

- **Função espacoDisponivel**

Outro dos principais desafios enfrentados foi o desenho da **casa**. Devido ao fato de a casa ocupar um espaço maior na matriz do quadro, logo ficou evidente que, ao tentar desenhar mais de 25 casas, não havia mais espaço disponível para posicionar novas figuras. Isso resultava em falhas na execução do programa ou em sobreposição de figuras.

A casa, com sua base larga e telhado triangular, ocupa uma quantidade significativa de posições na matriz, o que limita o número de vezes que pode ser desenhada sem sobrecarregar o quadro. Esse problema era agravado pelo uso de coordenadas aleatórias, pois, após várias iterações, muitas posições já estavam ocupadas, impossibilitando a inserção de novas casas.

Para resolver esse problema e melhorar a experiência do usuário, foram implementadas mensagens de aviso. Quando o programa detecta que não há mais espaço disponível para desenhar novas figuras, ele interrompe o processo de geração e exibe uma mensagem informando o usuário que o quadro está cheio.

Implementação:

- A função `espacoDisponivel()` percorre a matriz antes de cada tentativa de desenhar uma nova casa. Caso não haja mais espaço suficiente, o programa exibe uma mensagem como "Não há mais espaço disponível para desenhar novas figuras" e a execução é abortada.

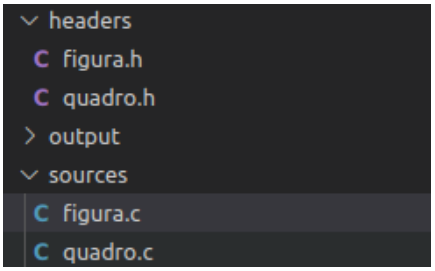
```
Digite o tipo de figura desejada: 5
Digite a quantidade de figuras (menor ou igual a zero para aleatório): 26
Não há mais espaço disponível para desenhar novas figuras.
```

A mensagem alerta que, ao tentar desenhar mais de 25 figuras, não há espaço suficiente no quadro

4. Compilação

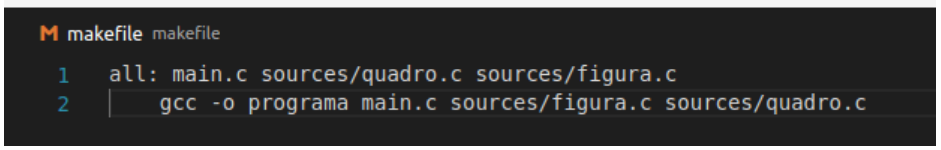
Para compilar o projeto, foi adotada uma organização em que os arquivos de código foram divididos em diferentes TADs (Tipos Abstratos de Dados) para garantir uma melhor organização e clareza no desenvolvimento. A estrutura está dividida em dois componentes principais:

- **quadro.h**: Contém a definição do quadro, as funções responsáveis por inicializar e imprimir o quadro, bem como a verificação de áreas disponíveis para desenhar as figuras.
- **figura.c**: O arquivo `figura.c` contém a implementação de funções responsáveis pela criação e manipulação das figuras geométricas usadas no projeto. Em sua estrutura, a função `criarFigura`, inicializa uma estrutura do tipo `Figura`. Essa função recebe três parâmetros: o tipo da figura, e suas coordenadas `x` e `y` no plano do quadro.



Divisão das pastas do projeto

Além dessa separação dos arquivos, foi criado um Makefile para simplificar o processo de compilação. O Makefile contém todas as instruções necessárias para compilar os arquivos do projeto de forma eficiente. Antes de executá-lo, verifique se o terminal está na pasta onde o Makefile está localizado, pois erros relacionados à localização de diretórios podem ocorrer caso ele esteja em outro local.

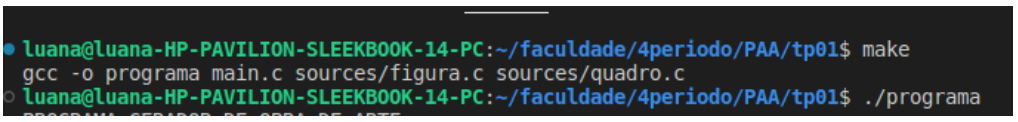


Estrutura do arquivo Makefile

Para compilar o programa, basta executar os comandos:

1- “make”

2- “./programa”



Comando utilizado na compilação do trabalho

5. Considerações finais

Durante o desenvolvimento deste trabalho, surgiram algumas dificuldades e facilidades na implementação das estruturas necessárias para a geração das "obras de arte" aleatórias. A principal dificuldade enfrentada foi a gestão do espaço disponível para desenhar as figuras, especialmente ao tentar evitar sobreposições e garantir que todas as figuras pudessem ser desenhadas corretamente dentro dos limites do quadro. A implementação de verificações para informar o usuário sobre a falta de espaço, bem como o controle da quantidade de figuras geradas, exigiu um planejamento cuidadoso.

Além disso, a criação de figuras mais complexas, como a "casinha", apresentou um desafio adicional, pois exigiu um controle mais detalhado de coordenadas para garantir que os desenhos fossem renderizados corretamente e não extrapolassem os limites do quadro, especialmente em execuções com muitas figuras.

Por outro lado, eu já tinha familiaridade com a utilização de TADs e com a organização do código em arquivos headers e sources, o que facilitou a modularização do projeto. O uso de um Makefile também simplificou o processo de compilação, economizando tempo e permitindo foco nas áreas mais desafiadoras do trabalho.

6. Referências bibliográficas

1. **YOUTUBE.** *Curso de Programação C | Como imprimir uma pirâmide de asteriscos? (triângulo isósceles) | aula 315*. Disponível em: https://www.youtube.com/watch?v=Ji605daGLAw. Acesso em: 3 out. 2024.
2. **ZIVIANI, N.** *Projeto de Algoritmos: com Implementações em Pascal e C*. 3. ed. São Paulo: Thomson Learning, 2004.
3. **YOUTUBE.** AULA ALGORITMOS EM C. *Aula 32 - Matriz na linguagem C.* Disponível em: https://www.youtube.com/watch?v=tv0lRp0vm_w. Acesso em: 4 out. 2024.