Trabalho Prático

Igor Eduardo Martins Braga - 2022425671
Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil
igoreduardobraga@ufmg.br

1. Introdução

O problema proposto foi implementar um programa que construísse fractais a partir de sistemas de reescrita de strings (L-sistemas). Era necessário escrever os fractais em cada estágio, sendo eles 1, 2, 3 e 4. A partir das strings geradas pelo programa, foram feitos, por meio de programas externos, o desenho de cada estágio dos fractais.

2. Fases do Projeto e Implementação:

O programa foi desenvolvido a partir da linguagem C. Para a compilação utilizou-se o compilador GCC.

Para implementar todos os fractais, foi utilizado a estratégia de arquivos iterativos, no qual os caracteres de um estágio intermediário são gravados em um arquivo, que posteriormente é lido e processado para gerar um novo arquivo correspondente ao próximo estágio.

2.1 Funções importantes para a construção do código

removerEspacos()

Essa função tem como objetivo receber um vetor de caracteres (string) e percorrer todos os elementos dessa string em busca de espaços vazios, removendo-os. Essa função foi implementada para lidar com a entrada fornecida pelo usuário, uma vez que o usuário pode inserir espaços indesejados durante a leitura da entrada.

escreverAxioma()

Essa função tem a finalidade de abrir o arquivo chamado "arquivo_1" no modo de escrita e escrever o axioma nele. Em seguida, ela abre o mesmo arquivo no modo de leitura, para que possa ser processado pelas funções específicas de cada fractal. Essa função foi desenvolvida porque é necessário escrever o axioma

no arquivo antes de processá-lo durante a execução do código de cada fractal. Dessa forma, a fim de permitir a reutilização dessa ferramenta para todos os fractais, foi decidido criar essa função.

2.2 Implementação do Fractal de Ilha de Koch

O fractal é implementado em etapas, seguindo uma estratégia que envolve o uso de arquivos. Para isso, são criados três arquivos distintos, um arquivo final e dois arquivos auxiliares:

1. Arquivo final: Este arquivo é usado para armazenar cada estágio do fractal conforme é processado.
2. Arquivos auxiliares: Esses arquivos são utilizados para obter todos os estágios do fractal. O conteúdo de um arquivo auxiliar é lido, processado e escrito no outro arquivo auxiliar. Esse processo ocorre de forma alternada: primeiro, o arquivo auxiliar 1 é lido e processado, e o resultado é escrito no segundo arquivo auxiliar 2. Em seguida, o arquivo auxiliar 2 é lido e processado, e o resultado é escrito de volta no arquivo auxiliar 1. Esse ciclo se repete até que o estágio desejado do fractal seja alcançado.

Todo o processo descrito é executado através de um loop "for" que realiza três iterações, nas quais são obtidos os estágios 2, 3 e 4 do fractal. Nesse fractal em particular, o axioma corresponde ao primeiro estágio, portanto não é necessário realizar operações dentro do loop "for" para obter o estágio 1. Basta escrever o axioma no arquivo final como sendo o primeiro estágio.

O processamento dos arquivos auxiliares ocorre dentro desse loop, no qual eles são lidos e os símbolos "F" são substituídos de acordo com as regras até se chegar ao estágio desejado. O trecho de código responsável por esse processamento é o seguinte:

```
while ((caractere = fgetc(arquivo_1)) != EOF) {
    if(caractere=='F'){
        fputs(regra, arquivo_2);
    }
    else{
        fputc(caractere, arquivo_2);
    }
}
```

O arquivo auxiliar é lido caractere por caractere, e caso o caractere seja um símbolo "F", ele é substituído pela regra do fractal, e o resultado é escrito em outro arquivo auxiliar. Se o caractere não for um símbolo "F", ou seja, for um símbolo "+" ou "-", nenhuma alteração é feita, e o caractere é colocado no outro arquivo.

Para garantir que a abertura e fechamento dos arquivos ocorram corretamente, foram criadas duas verificações no código. A primeira determina qual arquivo TXT deve ser aberto em modo leitura para que seu conteúdo seja escrito no arquivo final. Isso é feito através do seguinte trecho de código:

```
if(i%2==0){
    fclose(arquivo_2);
    arquivo_2 = fopen("arq_aux2.txt", "r");
}
else if(i%2==1){
    fclose(arquivo_2);
    arquivo_2 = fopen("arq_aux1.txt", "r");
}
```

Note que a abertura de cada TXT depende da paridade da iteração do loop “for”, o que garante a propriedade de alternância explicada anteriormente.

Após isso, o conteúdo desse arquivo aberto em modo leitura é percorrido e copiado para o arquivo final, representando um estágio.

Em seguida, há outra verificação para a abertura e fechamento de arquivos. Desta vez, determina-se qual arquivo deve ser aberto em modo leitura e qual arquivo deve ser aberto em modo escrita. Isso é determinado pelo seguinte trecho de código:

```
if(i%2==0){
    arquivo_1 = fopen("arq_aux2.txt", "r");
    arquivo_2 = fopen("arq_aux1.txt", "w");
}
else if(i%2==1){
    arquivo_1 = fopen("arq_aux1.txt", "r");
    arquivo_2 = fopen("arq_aux2.txt", "w");
}
```

Novamente, a paridade da iteração é utilizada para garantir a propriedade de alternância. Observe que essa parte do código indica a finalização do processamento de um estágio e determina qual arquivo receberá o novo estágio e qual será lido, para que essa operação seja feita. De acordo com a imagem acima, se a iteração for par, o arquivo auxiliar 2 será aberto em modo leitura, pois ele foi o

último a receber o conteúdo de um estágio e deve ser lido e processado para produzir o próximo estágio. Por outro lado, o arquivo auxiliar 1 será aberto em modo escrita para receber o próximo estágio do fractal. Essa lógica então é invertida a cada iteração por meio dessa verificação.

Por fim, os arquivos auxiliares são excluídos, mantendo apenas o arquivo final, que contém os quatro estágios completos do fractal.

2.3 Implementação do Fractal de Espaço de Peano

A implementação desse fractal é feita com a mesma ideia utilizada para o fractal de Ilha de Koch.

Utilizamos a estratégia de arquivos novamente, mas foram feitas pequenas alterações no código de Ilha de Koch para produzir o de Espaço de Peano. Elas podem ser listadas abaixo:

1. Loop “for” com uma iteração a mais: diferentemente do fractal de Ilha de Koch, o fractal de Espaço de Peano não tem como primeiro estágio o axioma. Logo, para alterar isso no código, o loop “for” é percorrido 4 vezes para determinar os 4 estágios desse fractal (de i=1 até i=5).
2. Substituição de símbolos: como o fractal de espaço de peano possui duas regras, uma com o símbolo X e outra com o símbolo Y, foi necessário adicionar condicionais para substituir esses símbolos pelas suas regras. Isso é feito por meio do seguinte código:

```
while ((caractere = fgetc(arquivo_1)) != EOF) {
    // Troca o caractere "X" pela regra do fractal e escreve o conteúdo resultate no outro arquivo auxiliar
    if(caractere=='X'){
        fputs(regra1, arquivo_2);
    }
    // Troca o caractere "Y" pela regra do fractal e escreve o conteúdo resultate no outro arquivo auxiliar
    else if(caractere=='Y'){
        fputs(regra2, arquivo_2);
    }
    else{
        fputc(caractere, arquivo_2);
    }
}
```

3. Retirar os símbolos “X” e “Y” de cada estágio ao acabar de processá-los: para isso, ao escrever o conteúdo do arquivo auxiliar que possui o estágio atual no arquivo final, ignora-se esses símbolos, escrevendo somente os F’s.
4. Verificação de abertura e fechamento de arquivos: como uma nova iteração foi adicionada ao loop “for” (iteração i=1), foi necessário alterar as condicionais das verificações. A primeira condicional passou a ser referente

às iterações ímpares, já que a iteração i=1, que é ímpar, foi adicionada ao loop “for”.

2.4 Implementação do Fractal definido por mim

A implementação desse fractal é idêntica ao do Espaço de Peano. A única alteração são as informações (axioma, regras e ângulo) colocadas no arquivo final, que são referentes a esse novo fractal produzido. Além disso, passamos por parâmetro para o código do novo fractal o seu axioma e regras, não sendo necessário ler essas informações a partir do teclado.

Para definir as regras, axiomas e grau para esse novo fractal, usou-se o fractal Espaço de Hilbert como inspiração. Alterou-se alguns símbolos nas regras desse fractal, removendo alguns e adicionando outros. Assim, chegou-se ao seguinte resultado:

Axioma: X+X+X+X+X+X

Ângulo: 60

Regras: X = -YF+X+FY-

Y = +XF-X-FX+

Onde “=” representa gera, ou seja, X gera “-YF+X+FY-” e Y gera “+XF-X-FX+”

OBS: Para escolher o fractal definido por mim, basta colocar qualquer número diferente de 5 e 7 (definem os fractais Ilha de Koch e Espaço de Peano respectivamente) ao rodar o programa.

3. Estratégias para implementar os fractais

Existiam várias estratégias possíveis para implementar o trabalho. Entre elas, destacaram-se três abordagens principais:

Versão Iterativa com Uso de Arquivos: Nessa abordagem, os caracteres de um estágio intermediário são gravados em um arquivo, que posteriormente é lido e processado para gerar um novo arquivo correspondente ao próximo estágio. Esse processo é repetido até chegar ao estágio desejado. Optar por essa estratégia oferece uma implementação mais intuitiva e simples. No entanto, demanda a manipulação de arquivos e uma estrutura de repetição para ler e escrever os dados, tornando o código potencialmente mais extenso.

Versão Recursiva: Nessa abordagem, os caracteres do estágio desejado são gerados de forma recursiva. No entanto, implementar essa versão requer um

maior conhecimento sobre o funcionamento da recursão e a definição de um ponto de parada ou condição de parada. Embora a versão recursiva resulte em um código mais condensado e eficiente, comparado à implementação utilizada, pois evita processos repetidos, pode ser menos intuitiva para entender o passo a passo da recursão. Essa abordagem é mais adequada ao problema proposto, pois a construção dos fractais se baseia na recursão.

Versão Iterativa com uso de Vetor de Char: Nessa abordagem, um vetor de caracteres (string) é diretamente modificado utilizando ponteiros e a biblioteca "string.h" para realizar a substituição dos elementos de acordo com as regras estabelecidas a partir do axioma. Essa implementação é mais complexa, pois exige conhecimentos mais específicos sobre a manipulação de ponteiros e funções para substituição. Além disso, o uso de múltiplas iterações nos vetores de caracteres pode aumentar a complexidade do código, tornando-o potencialmente menos eficiente. Portanto, essa abordagem não apresenta benefícios substanciais em relação à implementação utilizada.

Após considerar essas diferentes estratégias, optou-se por utilizar a primeira abordagem, a versão iterativa com uso de arquivos. Essa escolha se baseou na sua maior intuitividade e facilidade de implementação em comparação com as outras estratégias. Além disso, essa abordagem permite a visualização clara de cada estágio do fractal, uma vez que cada estágio é lido e armazenado em um arquivo separado. Uma desvantagem em relação à versão recursiva é que o código pode se tornar maior devido ao uso de estruturas de repetição como "while" e "for" para processar os estágios até chegar ao estágio desejado.

4. Equações de recorrência dos fractais

Para obter as equações de recorrência dos fractais, expandimos o número de símbolos em cada estágio e examinamos se havia algum padrão entre eles. Esse padrão foi expresso como uma equação matemática para facilitar a compreensão do que ocorre em cada iteração do programa.

4.1 Equação de recorrência do fractal Ilha de Koch

$T(n) = 4 * 9^n + f(n)$

$f(n) = 4 * 6 * 9^{(n-1)} + f(n-1)$

$f(0) = 3$

Considere que a equação de recorrência completa seja T(n), já que 4 * 9^n indica a quantidade de símbolos “F” e f(n) indica a quantidade de símbolos “+” e “-” no resultado final. A partir dessa equação de recorrência, podemos calcular a quantidade total de símbolos em cada iteração do código:

Iteração	# F	# +/-	# Símbolos Total
0	4	3	7
1	36	27	63
2	324	243	567
3	2916	2187	5103
…	…	…	…
n	4 * 9^n	f(n)	T(n)

OBS: No fractal da Ilha de Koch, o primeiro estágio é o axioma. Para determinar a quantidade de símbolos em cada estágio, começamos com n=0, que representa o primeiro estágio. O segundo estágio é determinado por n=1 e assim por diante.

4.2 Equação de recorrência do fractal Espaço de Peano

Equação considerando todos os símbolos

T(n)=S(n)+F(n)+X(n)+Y(n)

S(n)=X(n-1)*4+Y(n-1)*4+S(n-1)

F(n)=X(n-1)*8+Y(n-1)*8+F(n-1)

X(n)=X(n-1)*5+Y(n-1)*4

Y(n)=X(n-1)*4+Y(n-1)*5

X(0)=1

Y(0)=0

F(0)=0

S(0)=0

Equação sem os símbolos F

T(n)=S(n)+X(n)+Y(n)

S(n)=X(n-1)*4+Y(n-1)*4+S(n-1)

X(n)=X(n-1)*5+Y(n-1)*4

Y(n)=X(n-1)*4+Y(n-1)*5

X(0)=1

Y(0)=0

S(0)=0

F(x) é a equação que determina o número de F's na equação final, X(n) determina a quantidade de X's, Y(n) determina a quantidade de Y's e S(n) determina a quantidade de símbolos +/-. Considere que T(n) seja a equação que calcula a quantidade total de símbolos em cada iteração.

A partir dessas duas equações apresentadas, podemos calcular a quantidade desses símbolos em cada iteração, como mostrado na tabela abaixo:

Iteração	# F	# X	# Y	# +/-	# Símbolos Total
1	8	5	4	4	21
2	80	41	40	40	201
3	728	365	364	364	1821
4	6560	3281	3280	3280	16401
…	…	…	…	…	…
n	F(n)	X(n)	Y(n)	S(n)	T(n)

4.3 Equação de recorrência para o fractal definido por mim

Equação considerando todos os símbolos

T(n)=S(n)+F(n)+X(n)+Y(n)

S(n)=X(n-1)*4+Y(n-1)*4+S(n-1)

F(n)=X(n-1)*2+Y(n-1)*2+F(n-1)

X(n)=X(n-1)+Y(n-1)*3

Y(n)=X(n-1)*2

X(0)=6

Y(0)=0

F(0)=0

S(0)=5

Equação sem os símbolos F

T(n)=S(n)+F(n)+X(n)+Y(n)

S(n)=X(n-1)*4+Y(n-1)*4+S(n-1)

X(n)=X(n-1)+Y(n-1)*3

Y(n)=X(n-1)*2

X(0)=6

Y(0)=0

S(0)=5

F(x) é a equação que determina o número de F's na equação final, X(n) determina a quantidade de X's, Y(n) determina a quantidade de Y's e S(n) determina a quantidade de símbolos +/-. Considere que T(n) seja a equação que calcula a quantidade total de símbolos em cada iteração.

A partir dessas duas equações apresentadas, podemos calcular a quantidade desses símbolos em cada iteração, como mostrado na tabela abaixo:

Iteração	# F	# X	# Y	# +/-	# Símbolos Total
1	12	6	12	29	59
2	48	42	12	101	203
3	156	78	84	317	635
4	480	330	156	965	1931
…	…	…	…	…	…
n	F(n)	X(n)	Y(n)	S(n)	T(n)

5. Análise de complexidade dos fractais

Para fazer essa análise, consideramos a equação de recorrência de cada fractal.

5.1 Análise de complexidade do fractal Ilha de Koch

Vamos analisar a complexidade da equação do fractal de Ilha de Koch:

f(n) = 4 * 6 * 9^(n-1) + f(n-1)

Para entender melhor a relação entre f(n) e f(n-1), podemos expandir essa recursão:

f(n) = 4 * 6 * 9^(n-1) + (4 * 6 * 9^(n-2) + f(n-2))
= 4 * 6 * 9^(n-1) + 4 * 6 * 9^(n-2) + f(n-2)
= 4 * 6 * 9^(n-1) + 4 * 6 * 9^(n-2) + (4 * 6 * 9^(n-3) + f(n-3))
= ...

Podemos observar que a recursão de f(n) forma uma série geométrica comum, onde cada termo é multiplicado por 9 e adicionado ao próximo termo. O primeiro termo é 4 * 6 * 9^(n-1), o segundo termo é 4 * 6 * 9^(n-2), o terceiro termo é 4 * 6 * 9^(n-3), e assim por diante. A fórmula geral para a soma de uma série geométrica é:

S = a * (r^n - 1) / (r - 1),

onde "a" é o primeiro termo, "r" é a razão e "n" é o número de termos. No nosso caso, o primeiro termo é 4 * 6 * 9^(n-1) e a razão é 9. O número de termos é "n". Portanto, a soma total da série é:

f(n) = (4 * 6 * 9^(n-1) * (9^n - 1)) / (9 - 1)
= 24 * 9^n - 24

Agora, vamos analisar a complexidade da função original T(n):

T(n) = 4 * 9^n + f(n)

Substituindo o valor de f(n) que calculamos:

T(n) = 4 * 9^n + (24 * 9^n - 24)
= 28 * 9^n - 24

Assim, a complexidade da equação T(n) é O(9^n), porque o termo dominante é 9^n. Isso significa que o tempo de execução da equação T(n) cresce exponencialmente em relação ao valor de "n".

5.2 Análise de complexidade do fractal Espaço de Peano

Para fazer a análise de complexidade desse fractal, primeiro, vamos simplificar as expressões de S(n), F(n), X(n) e Y(n) usando a recursão:

S(n) = 4X(n-1) + 4Y(n-1) + S(n-1) = 4(5X(n-2) + 4Y(n-2)) + 4(4X(n-2) + 5Y(n-2)) + S(n-2) = 36X(n-2) + 36Y(n-2) + S(n-2) = 36(5X(n-3) + 4Y(n-3)) + 36(4X(n-3) + 5Y(n-3)) + S(n-3) = 324X(n-3) + 324Y(n-3) + S(n-3) … = 36^n * X(0) + 36^n * Y(0) + S(0) = 36^n

F(n) = 8X(n-1) + 8Y(n-1) + F(n-1) = 8(5X(n-2) + 4Y(n-2)) + 8(4X(n-2) + 5Y(n-2)) + F(n-2) = 72X(n-2) + 72Y(n-2) + F(n-2) = 72(5X(n-3) + 4Y(n-3)) + 72(4X(n-3) + 5Y(n-3)) + F(n-3) = 648X(n-3) + 648Y(n-3) + F(n-3) … = 72^n * X(0) + 72^n * Y(0) + F(0) = 72^n

X(n) = 5X(n-1) + 4Y(n-1) = (5^2 * X(0))+(4* (5* Y(0)+4* X(0))) +(5* X(-1)+4* Y(-1)) … = (5^n * X(0))+(4* (5^(n+1)-1)/(5 -1)* Y(0)+((5^(n+1)-1)/(5 -1)-n)* X(0)) = (5^n * X(0))+(16* (5^(n+1)-1)* Y(0)+((16* (5^(n+1)-1)-16*n)* X(0))

Y(n) = 4X(n-1) + 5Y(n-1) = (4^2 * X(0))+(5* (4* Y(0)+5* X(0))) +(4* X(-1)+5* Y(-1)) … = (4^n * X(0))+(5* (4^(n+1)-1)/(4 -1)* Y(0)+((5*(4^(n+1)-1)/(4 -1)-n)* X(0)) = (4^n * X(0))+(20* (4^(n+1)-1)* Y(0)+((20*(4^(n+1)-1)-20*n)* X(0))

Então, podemos substituir essas expressões na equação original:

T(n)=S(n)+F(n)+X(n)+Y(n) = (36^n+72^n)+(16*(5^(n+1)-1)* Y(0)+(16*(5^(n+1)-1)-16*n)* X(0))+(20*(4^(n+1)-1)*Y(0)+(20*(4^(n+1)-1)-20*n)*X(0))=(36^n+72^n)+(320*(9^(n+1)-9*n))* X(0)+(80*(9^(n+2)-9*n))* Y(0)

Considerando apenas o termo que cresce mais rápido quando n aumenta, e ignorando os coeficientes constantes, temos que o termo que domina é o que tem a potência de base maior, ou seja, o termo com o fator de 4^n. Portanto, a complexidade da equação é: O(9^n)

5.3 Análise de complexidade do fractal definido por mim

Para fazer a análise de complexidade desse fractal, primeiro, vamos simplificar as expressões de S(n), F(n), X(n) e Y(n) usando a recursão:

S(n) = 4X(n-1) + 4Y(n-1) + S(n-1) = 4(X(n-2) + 3Y(n-2)) + 4(2X(n-2)) + S(n-2) = 16X(n-2) + 16Y(n-2) + S(n-2) = 16(X(n-3) + 3Y(n-3)) + 16(2X(n-3)) + S(n-3) = 64X(n-3) + 64Y(n-3) + S(n-3) … = 4^n * X(0) + 4^n * Y(0) + S(0) = 4^n * 6 + S(0)

F(n) = 2X(n-1) + 2Y(n-1) + F(n-1) = 2(X(n-2) + 3Y(n-2)) + 2(2X(n-2)) + F(n-2) = 8X(n-2) + 8Y(n-2) + F(n-2) = 8(X(n-3) + 3Y(n-3)) + 8(2X(n-3)) + F(n-3) = 32X(n-3) + 32Y(n-3) + F(n-3) … = 2^n * X(0) + 2^n * Y(0) + F(0) = 2^n * 6

X(n) = X(n-1) + 3Y(n-1) = X(0) + 3Y(0) + X(-1) + … + X(-n+1) = X(0) - n = 6 - n

Y(n) = 2X(n-1) = 2(X(0) - n+1) = -2n+14

Então, podemos substituir essas expressões na equação original:

T(n)=S(n)+F(n)+X(n)+Y(n) = (4^n * 6 + S(0)) + (2^n * 6) + (6 - n) - (2n+14) = (4^n * 6 - n -14)+ (S(0)+F(0))+(6* (4^n+1))

Considerando apenas o termo que cresce mais rápido quando n aumenta, e ignorando os coeficientes constantes, temos que o termo que domina é o que tem a potência de base maior, ou seja, o termo com o fator de 4^n. Portanto, a complexidade da equação é: O(4^n)

6. Opções de software para desenhar os fractais

Para desenhar os fractais gerados pelo programa, utilizou-se o site Online Math Tools (https://onlinemathtools.com/l-system-generator), que oferece uma ampla variedade de ferramentas para diferentes áreas da matemática. Especificamente, utilizou-se a opção de gerador de L-systems disponível no site. Essa ferramenta permite inserir o axioma e as regras dos fractais desejados, e em seguida, produz uma imagem do fractal correspondente.

Optamos por utilizar esse site devido à sua extensa gama de ferramentas disponíveis para desenhar fractais. Ele permite definir várias regras diferentes para os fractais (até 5), determinar o tamanho da imagem, baixar as imagens geradas gratuitamente e até mesmo personalizar as cores das imagens, entre outras opções.

Embora existam outros sites, como https://www.kevs3d.co.uk/dev/lsystems/ e http://piratefsh.github.io/p5js-art/public/lsystems/, que também geram imagens e permitem a definição de várias regras diferentes para os fractais, eles não possuem, por padrão, a função de download das imagens geradas. Portanto, a online Math Tools se mostrou a melhor opção devido à sua capacidade de fornecer todas as funcionalidades necessárias para a construção desse trabalho.

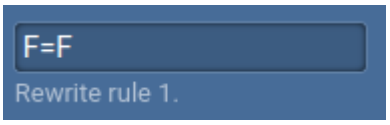
Como dito anteriormente, para verificar as strings geradas pelo programa, utilizamos o site Online Math Tools. Definimos a primeira regra como "F=F", o que significa que a letra F se substitui por ela mesma. Dessa forma, inserimos a string de saída na área do axioma e a imagem correspondente a essa string é desenhada.

Essa abordagem é possível devido à regra que estabelecemos. Cada ocorrência da letra F na string é substituída de forma idêntica por outra letra F, o que não altera a string original fornecida como axioma. Assim, o site reproduz diretamente a imagem do L-system desejado com base na string de saída do programa.

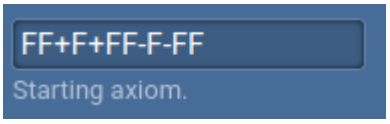
Em resumo, ao definirmos a regra "F=F" no site Online Math Tools e fornecermos a string resultante como axioma, a imagem do L-system desejado é desenhada com precisão, pois a regra assegura que cada ocorrência da letra F seja substituída por outra letra F na mesma quantidade, preservando assim a estrutura original da string. Para ilustrar isso, segue o passo a passo:

Esse passo a passo vai ser feito com o fractal de Espaço de Peano

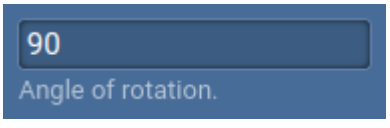
1. Defina a primeira regra como F=F:



2. Defina o axioma como sendo a saída do programa. Por exemplo, a string gerada para o primeiro estágio de espaço de Peano é definida por “FF+F+FF-F-FF”. Logo, ela será colocada como axioma:



3. Defina o ângulo do fractal

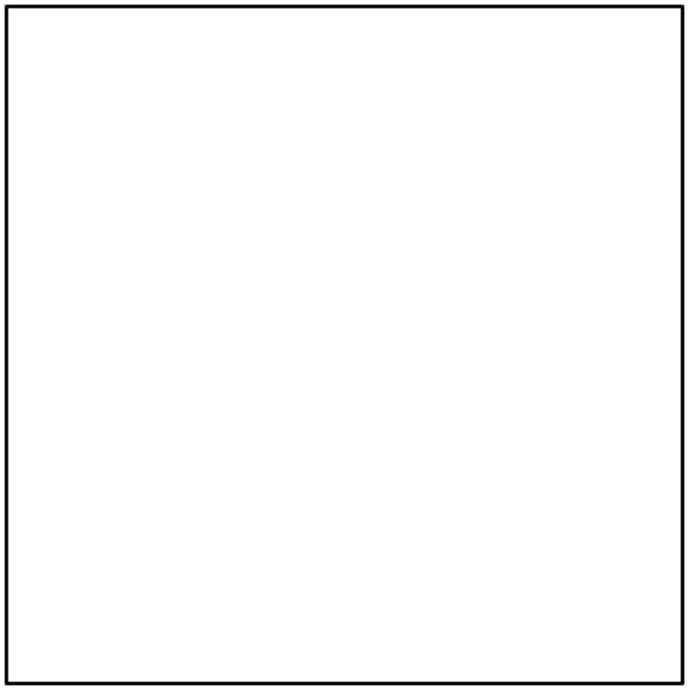


Outras opções, como o tamanho e cores da imagem, espessura da linha do fractal, entre outros, ficam a cargo do usuário.

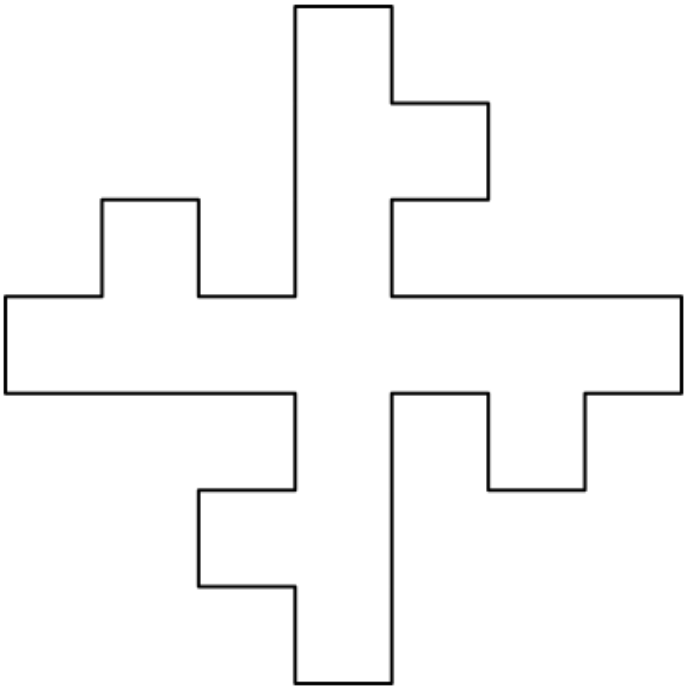
Imagens de cada estágio dos fractais:

Ilha de Koch

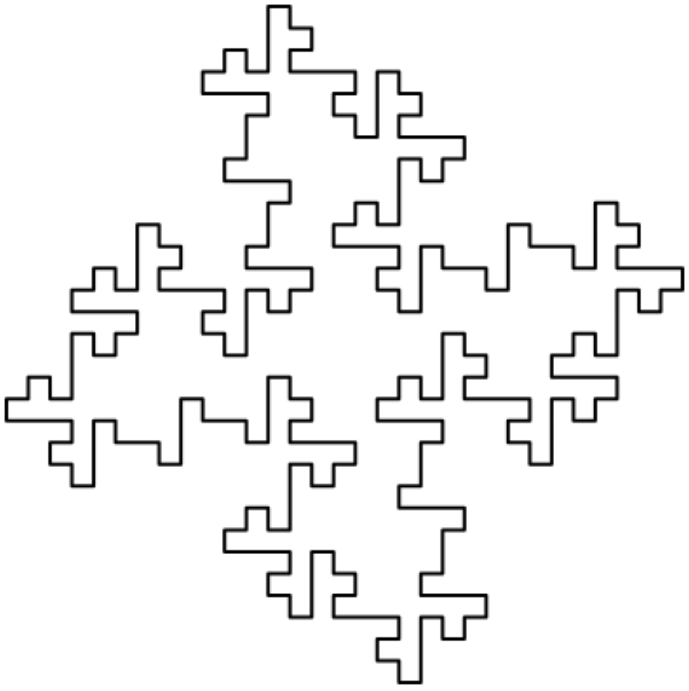
Primeiro estágio



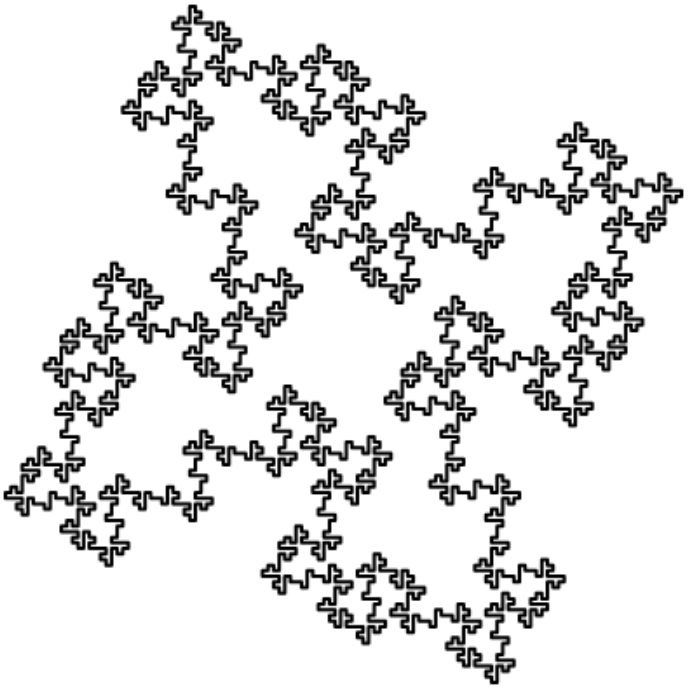
Segundo estágio



Terceiro estágio

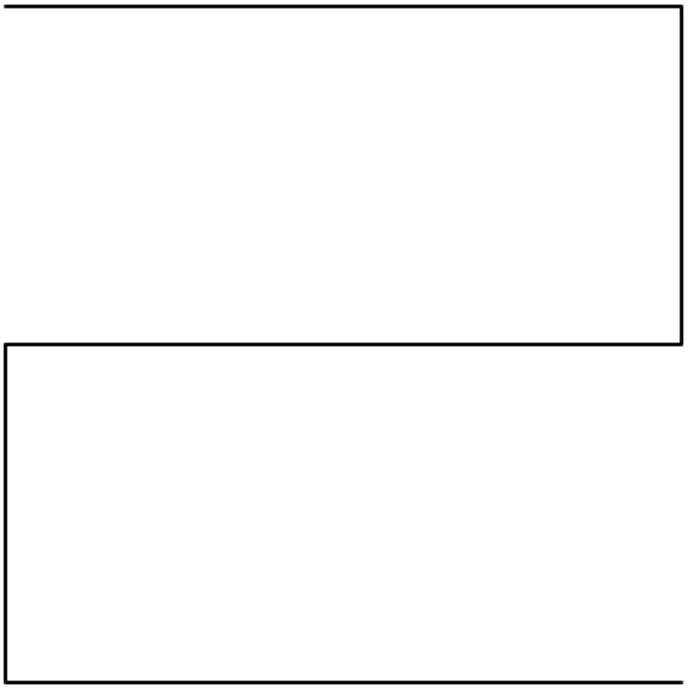


Quarto estágio

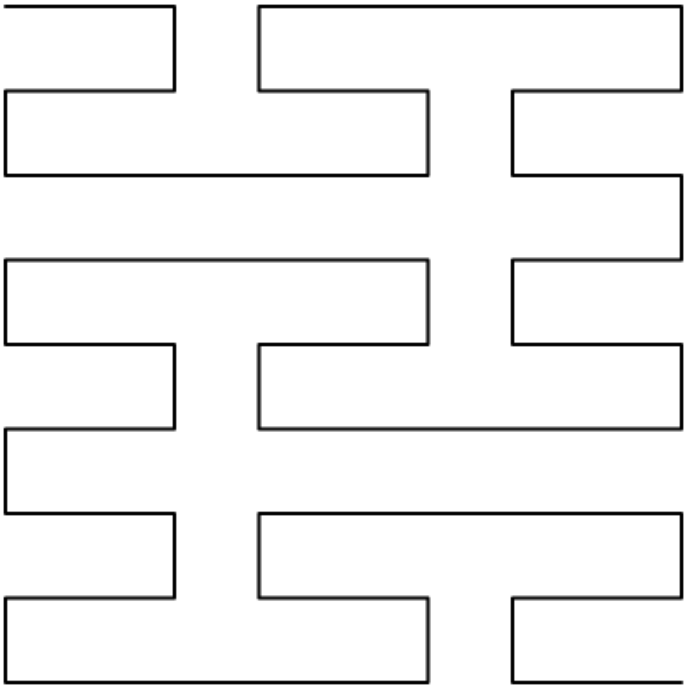


Espaço de Peano

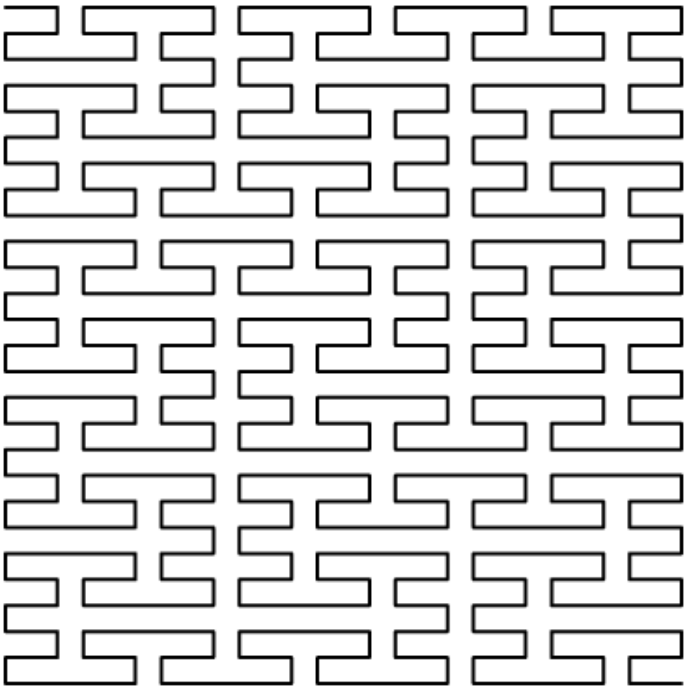
Primeiro estágio



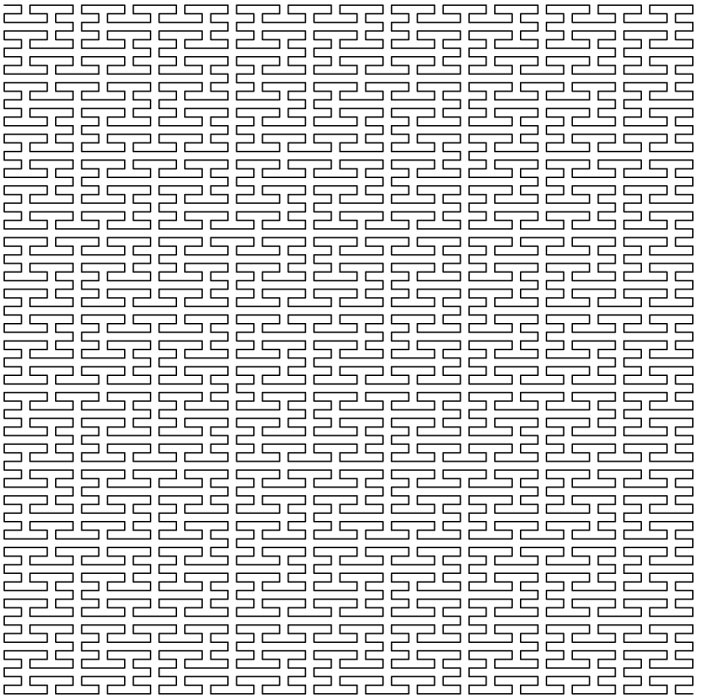
Segundo estágio



Terceiro estágio

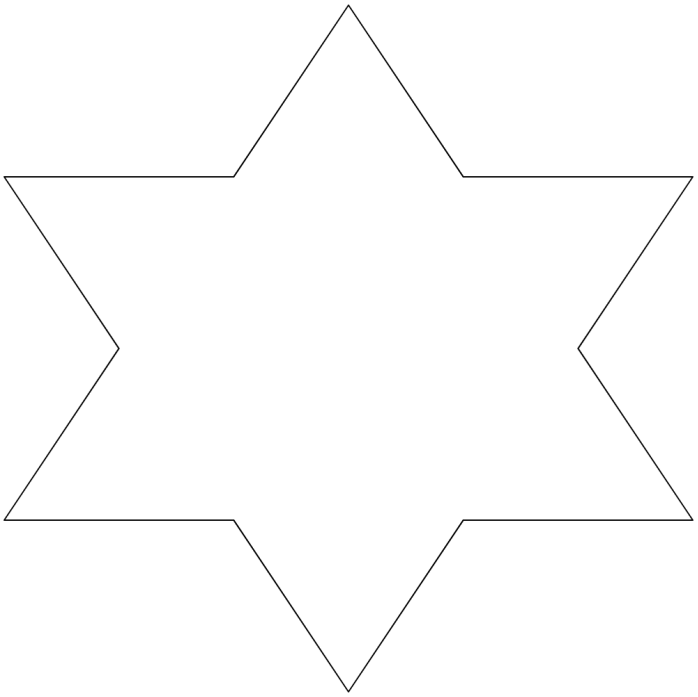


Quarto estágio

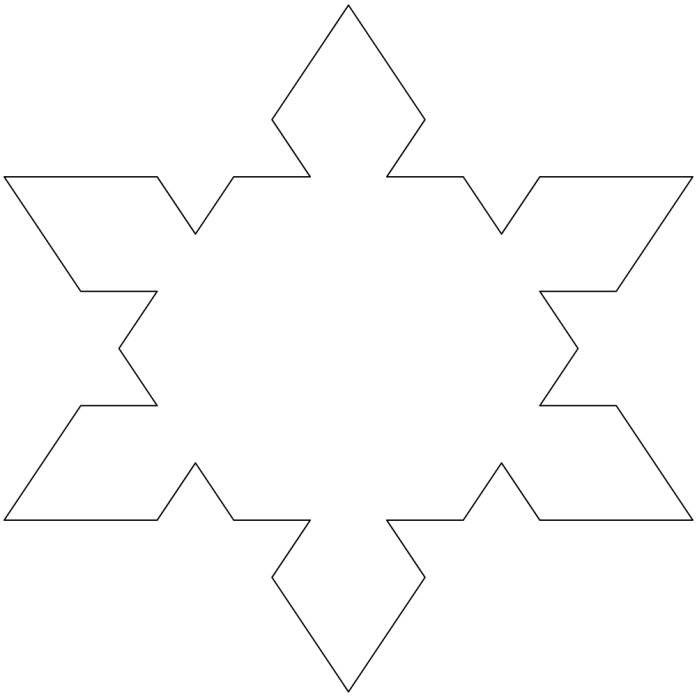


Fractal definido por mim

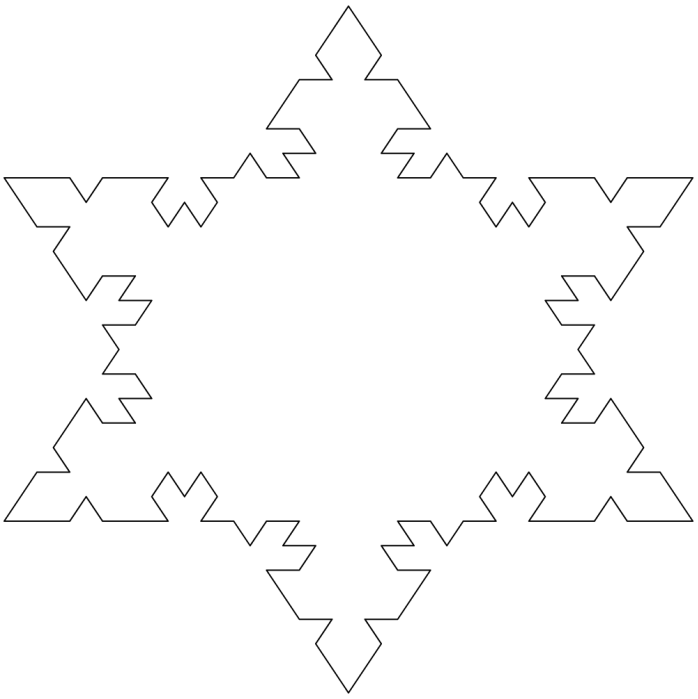
Primeiro estágio



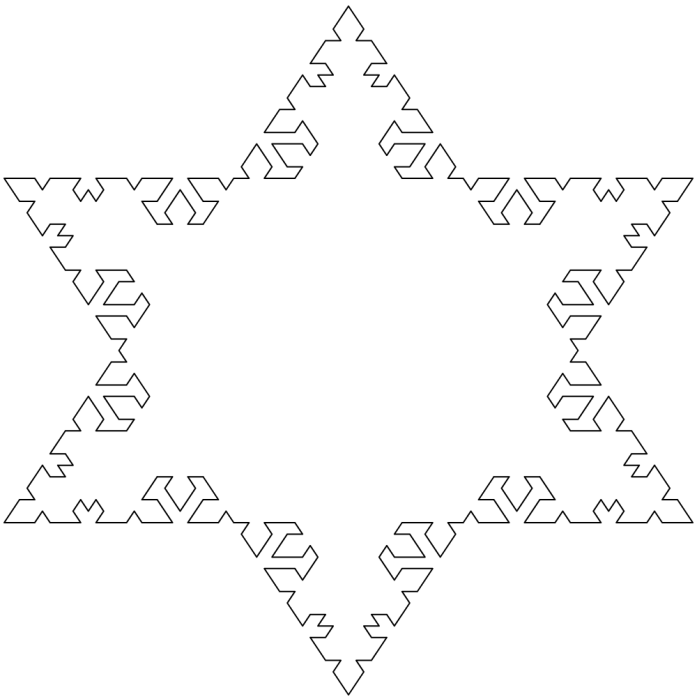
Segundo estágio



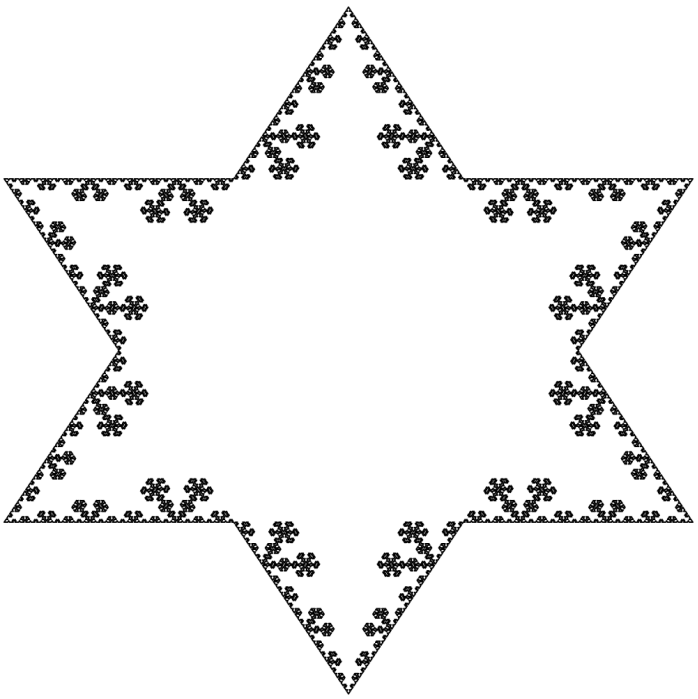
Terceiro estágio



Quarto estágio



Sétimo estágio



7. Conclusão

Durante a realização deste trabalho, adquiriu-se um amplo conhecimento sobre fractais e processos recursivos. Essa experiência foi extremamente valiosa para aprofundar nosso entendimento sobre recursão, equações de recorrência e como realizar a análise de complexidade a partir dessas equações. Além disso, o trabalho proporcionou uma maior compreensão sobre a utilização de arquivos em programas desenvolvidos na linguagem C.