MAC110 Introdução à Computação
EP no. 2

Prof. Dr. Paulo Miranda
Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP)

Um fractal é um objeto geométrico que pode ser dividido em partes, cada uma das quais semelhante ao objeto original. Fractais são muito usados em arte gerada por computador. O objetivo deste EP é fazer programas em linguagem C, para gerar imagens (nos formatos PGM e PPM) com desenhos derivados do conjunto de Mandelbrot.

Conjunto de Mandelbrot:

Em matemática, conjunto de Mandelbrot é um fractal definido como o conjunto de pontos c no plano complexo para o qual a sequência definida pela seguinte fórmula de recorrência:

$z_0 = 0$
$z_{n+1} = z_n^2 + c$

não tende ao infinito.

O conjunto de Mandelbrot, em sua representação gráfica no espaço contínuo, pode ser dividido em um conjunto infinito de figuras pretas, sendo a maior delas um cardióide localizado ao centro do plano complexo. Existe uma infinidade de quase-círculos que tangenciam o cardióide e variam de tamanho com raio tendendo assintoticamente a zero. Cada um desses círculos tem seu próprio conjunto infinito de pequenos círculos cujos raios também tendem assintoticamente a zero. Esse processo se repete infinitamente, gerando uma figura fractal.

A) Desenhando o conjunto:

No espaço discreto de uma imagem digital, a repetição da figura fractal é interrompida a partir de uma certa altura devido a limitada resolução da imagem, sendo que apenas um conjunto finito de pontos amostrados são visualizados (Figuras 1 e 2). No entanto, a quantidade de cálculos necessária para se gerarem tais imagens está além da capacidade de um ser humano executar a mão. O Professor Benoît Mandelbrot foi a primeira pessoa a utilizar um computador para plotar o conjunto.

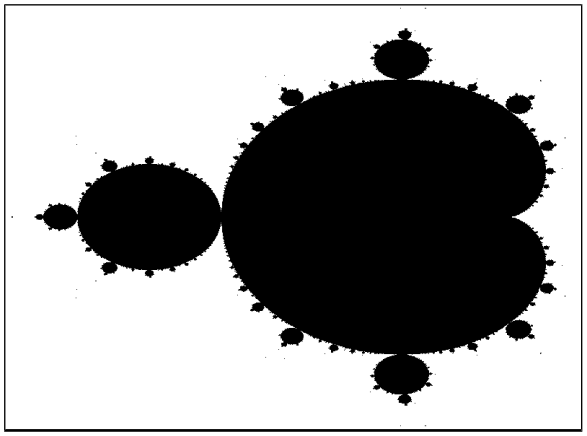


Figura 1: Conjunto de Mandelbrot (em preto), onde o canto inferior esquerdo corresponde ao ponto do plano de Argand-Gauss (-1.5, -1.0) e o canto superior direito é (0.5, 1.0).

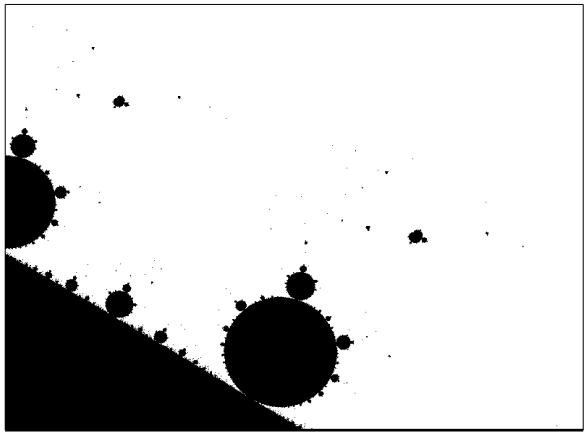


Figura 2: Conjunto de Mandelbrot (em preto), onde o canto inferior esquerdo corresponde ao ponto do plano de Argand-Gauss (0.278587, -0.012560) e o canto superior direito é (0.285413, -0.007440).

A sequência de pontos no plano complexo obtida pela fórmula de recorrência é chamada órbita de z_0 sob a transformação $z_{n+1} = z_n^2 + c$. É possível provar que uma vez que um ponto atinge uma distância da origem maior que 2 (valor absoluto de z_n maior que 2), a órbita explode para o infinito e, portanto, c não pertence ao conjunto de Mandelbrot. Este valor, conhecido como valor de fuga, permite o término dos cálculos para pontos que não pertencem ao conjunto de Mandelbrot. Para aqueles que pertencem ao conjunto, ou seja, valores de c para os quais z_n não tende ao infinito, o cálculo nunca irá terminar. Portanto, o cálculo deve ser terminado após um certo número de iterações determinado pelo programa. Isso resulta em uma imagem que é apenas uma representação aproximada do conjunto verdadeiro.

B) <u>Adicionando cor</u>:

Teoricamente, as imagens dos conjuntos de Mandelbrot são binárias (os pontos pertencem ou não ao conjunto), no entanto, as imagens de arte gerada por computador são desenhadas em cores.

No método de renderização mais comum, para os pontos que divergem para o infinito e que, portanto, não fazem parte do conjunto, as cores refletem o número de iterações necessárias para atingirem o valor de fuga. Isso cria formas concêntricas, cada uma sendo uma melhor aproximação para o conjunto de Mandelbrot em relação à camada exterior (Figuras 3 e 4).

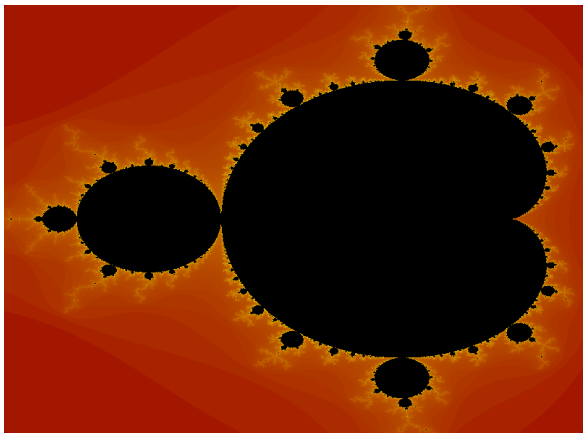


Figura 3: Renderização em cores da Figura 1, onde o conjunto de Mandelbrot é representado em preto, e pontos fora do conjunto são coloridos de acordo com sua velocidade de escape.

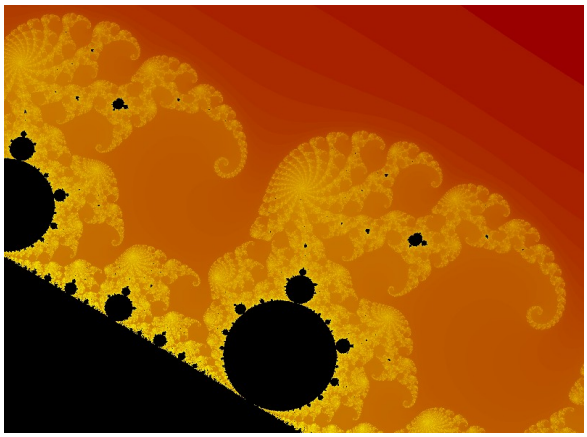


Figura 4: Renderização em cores da Figura 2, onde o conjunto de Mandelbrot é representado em preto, e pontos fora do conjunto são coloridos de acordo com sua velocidade de escape.

Para um conjunto finito A de pontos amostrados para visualização, seja $A_F \subset A$ o subconjunto de pontos que não pertencem ao conjunto de Mandelbrot (subconjunto dos pontos de fuga). Para um dado ponto $c \in A_F$, seja $k(c)$ o seu número de iterações de fuga, e k_{min} e k_{max} o menor e maior número de iterações de fuga encontrados no conjunto A_F, respectivamente.

$$k_{max} = \max_{c \in A_F} k(c)$$
$$k_{min} = \min_{c \in A_F} k(c)$$

Dadas duas cores C1 e C2, um esquema possível é pintar de C1 os pontos $c \in A_F$ que divergem rapidamente (isto é, $k(c) = k_{min}$), e de C2 os que divergem mais vagarosamente (isto é, $k(c) = k_{max}$). As cores para pontos com números intermediários de iterações (isto é, $k_{min} < k(c) < k_{max}$) são então obtidas através da média ponderada de C1 e C2, gerando um gradiente de cores (cores interpoladas).

Visto que o número de iterações de fuga para pontos próximos ao conjunto de Mandelbrot pode ser extremamente elevado, em comparação com os demais pontos, uma estratégia de realce para melhorar a visualização consiste em aplicar a função logarítmica no cálculo dos pesos, usando a seguinte fórmula:

$$p = \frac{\log(k(c) - k_{min} + 1)}{\log(k_{max} - k_{min} + 1)}$$

onde p é o peso da cor C2, e $q = (1.0 - p)$ é o peso da cor C1 na média ponderada.

C) <u>Buddhabrot</u>:

O Buddhabrot é uma representação gráfica especial do conjunto de Mandelbrot (Figura 5) que, quando girada em 90 graus no sentido horário, se assemelha a algumas representações do Buda (em inglês Buddha). Para gerar a representação de Buddhabrot usamos um vetor bidimensional de contadores, um para cada pixel da tela. Então um conjunto A de pontos de amostragem é iterado na função de Mandelbrot. Para os pontos $c \in A$ que escapam em um dado número de iterações (ou seja, pontos do subconjunto $A_F \subset A$, que não fazem parte do conjunto de Mandelbrot), a órbita da transformação correspondente, obtida pela fórmula de recorrência, é desenhada sobre o mapa de contadores. Mais precisamente, os contadores correspondentes as posições dos pontos z_n, de cada iteração da sequência no plano complexo, são incrementados para cada vez que z_n for igual a posição do contador.

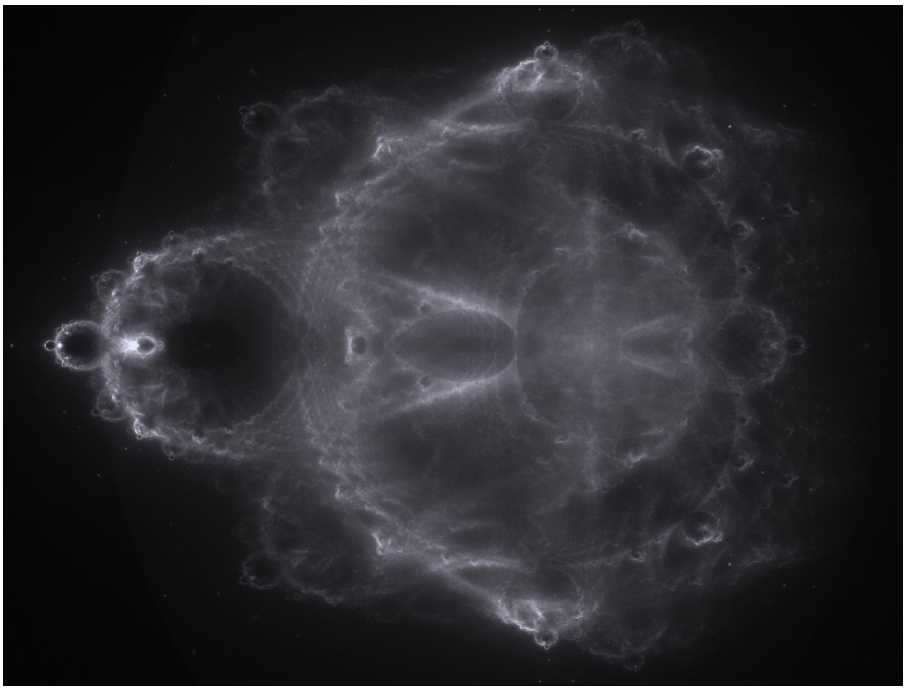


Figura 5: Buddhabrot com 20.000 iterações, onde o canto inferior esquerdo corresponde ao ponto do plano de Argand-Gauss (-1.5, -1.0) e o canto superior direito é (0.8, 1.0).

Após a iteração sobre um número grande de pontos $c \in A$, as cores dos pixels são escolhidas com base nos valores de cada contador. Um esquema possível é pintar da cor C1 os pixels com contagem mínima C_{min}, e da cor C2 os que possuem contagem máxima C_{max}. As cores para pixels com valores intermediários de contagem (isto é, $C_{min} < count < C_{max}$) são então obtidas através da média ponderada de C1 e C2, gerando um gradiente de cores (cores interpoladas). O cálculo dos pesos é dado pela seguinte equação:

$$p = \frac{count - C_{min}}{C_{max} - C_{min}}$$

onde p é o peso da cor C2, e $q = (1.0 - p)$ é o peso da cor C1 na média ponderada.

Uma estratégia de realce para melhorar a visualização consiste em aplicar a seguinte correção no peso p:

$$p_{novo} = min(p_{ant} \times 2.5,\ 1.0)$$

Atividade:

Faça programas em linguagem C que geram imagens dos desenhos dos três tipos de representações gráficas do conjunto de Mandelbrot, apresentadas anteriormente, até um valor máximo de iterações fornecido pelo usuário. O primeiro programa (**ep02_a.c**) deve gerar a imagem binária (valor 0 ou 255) do conjunto de Mandelbrot no formato PGM (arquivo **mandelbrot.pgm**). O segundo programa (**ep02_b.c**) deve gerar a renderização em cores do conjunto de Mandelbrot no formato PPM (arquivo **mandelbrot.ppm**), para quaisquer cores da sua preferência. O terceiro programa (**ep02_c.c**) deve gerar a imagem da representação de Buddhabrot (sem rotação) no formato PPM (arquivo **buddhabrot.ppm**), para quaisquer cores da sua preferência.

Os parâmetros dos programas são:

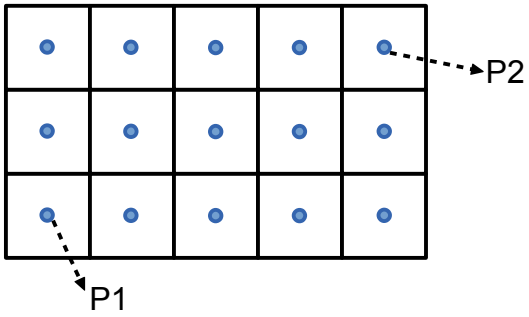
- Número máximo de iterações da fórmula de recorrência (ex: 10000, 20000),
- Coordenadas do canto inferior esquerdo (ponto P1) e do canto superior direito (ponto P2) da região visualizada do plano de Argand-Gauss.

Para o terceiro programa temos também o seguinte parâmetro adicional:

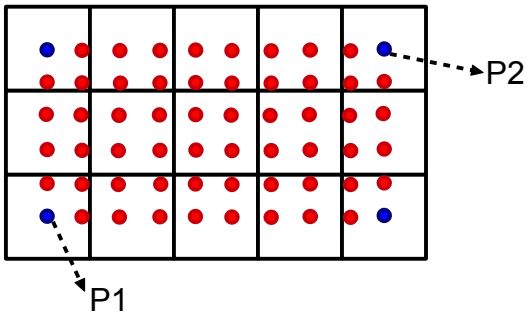
- Fator multiplicativo da quantidade de amostras por linha e coluna (ex: 2, 5, 10).

No caso dos problemas 1 e 2, temos que as amostras $c \in A$ correspondem aos centros dos pixels, tal que o

centro do pixel na coordenada linha e coluna (m-1,0) corresponde ao ponto do plano complexo P1 e o centro do pixel em (0,n-1) corresponde ao ponto do plano complexo P2.



No caso do terceiro programa, precisamos de um número maior de amostras em *A*, a fim de obter uma maior relevância estatística no acesso aos contadores, o que resulta em uma imagem final de Buddhabrot com maior qualidade. As amostras são selecionadas em intervalos igualmente espaçados, sendo o número total de amostras igual ao número total de pixels vezes o fator multiplicativo ao quadrado. O exemplo abaixo mostra um caso onde o fator multiplicativo é 2, ou seja temos o dobro de amostras em cada linha e coluna (5x2=10 amostras por linha e 3x2=6 amostras por coluna).



Observações:
No problema 2, para usar a função logarítmica (**double log(double x);**) você deve incluir a biblioteca math.h (**#include <math.h>**), e no comando de compilação do gcc acrescentar a opção **-lm**:

gcc ep02_b.c -o ep02_b -lm

Sem o **-lm** você vai obter a seguinte mensagem de erro:

/tmp/cc1akiRG.o: In function `main':
ep02_b.c:(.text+0x42f): undefined reference to `log'
ep02_b.c:(.text+0x45c): undefined reference to `log'
collect2: error: ld returned 1 exit status

Com relação ao tamanho das imagens, você pode usar valores como 640x480, 800x600, 1024x768, ou qualquer outro tamanho. No entanto, para valores grandes você deve declarar os vetores usando a classe de armazenamento **static** (estática):

```
#define MAX_W 1024
#define MAX_H  768

int main(){
   static int C[MAX_H][MAX_W];
   ...
   return 0;
}
```

As variáveis declaradas dentro da função **main**, sem o modificador **static**, são da classe de armazenamento **auto** (automática). As variáveis automáticas são alocadas na pilha. O tamanho máximo de pilha para um programa em C é dependente do sistema. Variáveis da classe estática são armazenadas na área de armazenamento estática, sendo mais recomendáveis para armazenar grandes quantidades de dados.