

1 Gráfico de polinômios (++++)



(++++)

Faça um programa que, dado um polinômio, de ordem máxima 100, apresente o seu gráfico em uma imagem colorida no formato PPM de dimensão máxima 300x300. O programa deve ler o polinômio desejado no formato: $y = a_0x^0 + a_1x^1 + a_2x^2 + \dots + a_nx^n$, onde a_i é o coeficiente do polinômio na ordem i . Por exemplo, polinômio “ $y = 2x^2 + 3x^1 + 4x^0$ ” define uma equação do segundo grau, com $a_0 = 4, a_1 = 3, a_2 = 2$. A imagem resultante deve ter dimensão $(2 \times N + 1)$, de fundo com cor (200,210,220) e eixos (x e y) na cor preta centrados no centro da imagem, com escala de S pixels por unidade. A função deve ser desenhada na cor azul (0,0,255).

Entrada

O programa deve iniciar com a leitura da ordem O , do polinômio e ler $O + 1$ coeficientes reais, iniciando pelo coeficiente de menor ordem até o de maior ordem. Após a leitura do polinômio o programa deve ler os valores de N e S .

Saída

Um texto que representa uma imagem PPM. Para o exemplo de uma imagem 101x101, a saída deve estar de acordo com a sequência:

```
P3
101 101
255
<dados da imagem>
```

Observações

O seu código deve ser construído a partir do código abaixo.

```
1 #include <stdio.h>
2
3 #define N 300          // Maxima dimensao da imagem
4 #define OM 100         // Maxima ordem do polinomio
5
6 /**
7  * Estrutura que armazena um polinomio
8  */
9 typedef struct {
10     int ordem;
11     float c[OM];
12 } Poly;
13
14 /**
15  * Estrutura que representa um ponto na imagem colorida
16  */
17 typedef struct {
18     int r, g, b;
19 } Pixel;
20
21 /**
22  * Estrutura que representa uma imagem colorida
23  */
```

```

24 typedef struct {
25     int largura, altura;
26     Pixel p[N][N];
27 } Image;
28
29 /**
30  * Funcao que le um polinomio e o retorna como uma estrutura Poly
31  */
32 Poly le_polinomio( void );
33
34 /**
35  * Funcao que recebe um polinomio e retorna o valor do polinomio para um
36  * determinado valor de x.
37  *
38  * @param p polinomio do tipo Poly
39  * @param x valor da variavel x
40  * @return o valor do polinomio para o valor de x
41  */
42 float calcula_polinomio( Poly p, float x);
43
44 // Imagem de trabalho (Variavel Global)
45 Image img;
46
47 /**
48  * Atribui o valor do pixel p a todos os pixels da imagem img.
49  * @param p Pixel contendo a cor de fundo desejada.
50  */
51 void preenche_fundo( Pixel p );
52
53 /**
54  * Funcao que desenha os eixos X e Y na imagem img.
55  */
56 void desenha_eixos( void );
57
58 /**
59  * Funcao que desenha a funcao do polinomio p na imagem img.
60  *
61  * @param p polinomio a ser desenhado.
62  */
63 void desenha_funcao( Poly p);
64
65 /**
66  * Funcao que imprime a imagem img no terminal.
67  * Para visualizar a imagem, voce deve redirecionar a saida padrao
68  * para um arquivo do tipo PPM no comando de execucao do programa.
69  * Por exemplo: ./programa > img.ppm <ENTER>
70  */
71 void print_image( void );
72
73
74 /**
75  * Programa principal. Este programa le um polinomio e o desenha em uma
76  * imagem no formato PPM.
77  */
78 int main() {
79
80     // ... [COLOQUE SEU CODIGO]
81
82     Poly poly; // Variavel que armazenara o polinomio
83     Pixel px;  // Variavel para armazenar a cor de fundo

```

```

84
85 // Leitura do polinomio
86 poly = le_polinomio();
87
88 // Leitura da dimensao da imagem
89 scanf("%d%d", &img.largura, &img.altura);
90
91 // Definicao da cor de fundo
92 px.r = 200;
93 px.g = 210;
94 px.b = 220;
95 // Chama a funcao que inicia o fundo da imagem.
96 // p e um Pixel que contem as cores do fundo.
97 preenche_fundo( px );
98
99 // Desenha os eixos X e Y com origem no centro da imagem.
100 desenha_eixos();
101
102 // ... [COLOQUE SEU CODIGO] ... Como considerar a ESCALA?
103 // Adapte o codigo para o uso da escala na funcao desenha_funcao()
104
105 // Desenha a funcao do polinomio p na imagem.
106 desenha_funcao( poly );
107
108 // Imprime a imagem final.
109 print_image();
110
111 return 0;
112 }
113
114 Poly le_polinomio( void ) {
115     int i;
116     Poly poly;
117     scanf("%d", &poly.ordem);
118
119     // ... [COLOQUE SEU CODIGO]
120
121     return poly;
122 }
123
124 float calcula_polinomio( Poly p, float x) {
125     // ... [COLOQUE SEU CODIGO]
126 }
127
128 void preenche_fundo( Pixel p ) {
129     int i, j;
130     for( i=0; i<img.altura; i++ ) {
131         for( j=0; j<img.largura; j++ ) {
132             // ... [COLOQUE SEU CODIGO]
133         }
134     }
135 }
136
137 void desenha_eixos( void ) {
138     int i, j;
139     Pixel p;
140     p.r = p.g = p.b = 0;
141
142     // calcula a metade da largura da imagem
143     j = img.largura / 2;

```

```

144 // desenha o eixo vertical
145 for( i = 0; i < img.altura; i++ ) {
146     img.p[i][ j ] = p;
147 }
148
149 // calcula a metade da altura da imagem
150 i = img.altura / 2;
151 // desenha o eixo horizontal
152 for( j = 0; j < img.largura; j++ ) {
153     img.p[i][ j ] = p;
154 }
155
156 }
157
158 void print_image( void ) {
159
160     int i, j;
161
162     printf("P3\n%d %d\n255\n", img.largura, img.altura);
163     for( i=0; i<img.altura; i++ ) {
164         for( j=0; j<img.largura; j++ ) {
165
166             // ... [COLOQUE SEU CODIGO]
167
168         }
169         printf("\n");
170     }
171 }
172
173 void desenha_funcao( Poly p ) {
174     // ... [COLOQUE SEU CODIGO]
175 }

```

main_partial.c