

2º Projeto de Cálculo Numérico (Interpolação Polinomial)

Prof^a. Vanessa Rolnik

Data de entrega: 07/05/2013

Danylo Augusto Pontes Goulart

nºUSP 6422120

Emerson Takeshi Urushibata

nºUSP 6402340

Código-fonte do programa em C:

```
1. #include <stdio.h>
2. #include <stdlib.h>
3.
4. // protótipo da função
5. int aprox(float pto, float mat[100][2], int total);
6.
7. int main(void) {
8.     FILE *fp;
9.     int total, i, j, apr;
10.    float mat[100][2], pto = -1,f,r;
11.    float x0,x1,x2,x3,aux;
12.    float o00,o01,o02,o03,o0aux;
13.    float o10,o11,o12,o1aux;
14.    float o20,o21,o2aux;
15.    float o30,o3aux;
16.    float o4aux;
17.
18.    // Abre o arquivo
19.    fp = fopen("dadoscap.txt", "r");
20.
21.    // Verifica se o arquivo dadoscap.txt existe
22.    if (fp == NULL) {
23.        printf("Erro: Não foi possível encontrar o arquivo
24.        \'dadoscap.txt\'\n");
25.        exit(0);
26.    } else {
27.        // Leitura da primeira linha (total de pontos coletados)
28.        fscanf(fp, "%d", &total);
29.        i = 0;
30.        // Executar a iteração até que os termos do arquivo
31.        dadoscap.txt acabem
32.        // Armazena os valores de t em mat[i][0] e V(t) em mat[i][1]
33.        while (!feof(fp)) {
34.            // Leitura das colunas do arquivo
35.            fscanf(fp, "%f %f", &mat[i][0], &mat[i][1]);
36.            i++;
37.        }
38.    }
39.    // Fecha o arquivo
40.    fclose(fp);
```

```

40.
41.     // Leitura do ponto de aproximação
42.     while (pto<0||pto>155) {
43.         printf("\nEntre com um ponto de aproximação no intervalo
[0,155]: ");
44.         scanf("%f", &pto);
45.         if (pto<0||pto>155)
46.             printf("Erro: O valor digitado está fora do intervalo
[0, 155].");
47.     }
48.
49.     // Chamada para a função que retorna o valor aproximado entre
os índices 0 e 30.
50.     // O retorno é um valor de 0 a 28, pois o grau 3, necessita
de 3 valores
51.     // pondendo ser os índices (0,1,2,3), (1,2,3,4), ...,
(28,29,30,31).
52.     apr = aprox(pto, mat, total);
53.
54.     // Cálculo da Diferença Dividida
55.     // x
56.     x0 = mat[apr][0];
57.     x1 = mat[apr+1][0];
58.     x2 = mat[apr+2][0];
59.     x3 = mat[apr+3][0];
60.
61.     // ordem 0
62.     o00 = mat[apr][1];
63.     o01 = mat[apr+1][1];
64.     o02 = mat[apr+2][1];
65.     o03 = mat[apr+3][1];
66.
67.     // ordem 1
68.     o10 = (o01-o00)/(x1-x0);
69.     o11 = (o02-o01)/(x2-x1);
70.     o12 = (o03-o02)/(x3-x2);
71.
72.     // ordem 2
73.     o20 = (o11-o10)/(x2-x0);
74.     o21 = (o12-o11)/(x3-x1);
75.
76.     // ordem 3
77.     o30 = (o21-o20)/(x3-x0);
78.
79.     // f aproximado
80.     f = o00 + (pto-x0)*o10 + (pto-x0)*(pto-x1)*o20 + (pto-
x0)*(pto-x1)*(pto-x2)*o30;
81.
82.     // Cálculo da aproximação de erro
83.     if (apr==28) {

```

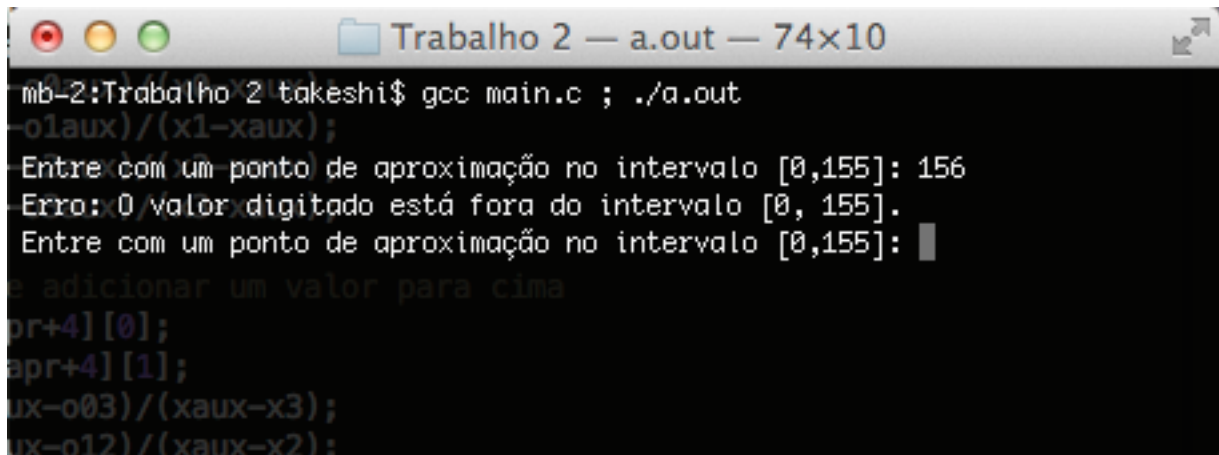
```

84.         // Caso tenha que adicionar um valor para baixo
85.         xaux = mat[apr-1][0];
86.         o0aux = mat[apr-1][1];
87.         o1aux = (o00-o0aux)/(x0-xaux);
88.         o2aux = (o10-o1aux)/(x1-xaux);
89.         o3aux = (o20-o2aux)/(x2-xaux);
90.         o4aux = (o30-o3aux)/(x3-xaux);
91.     } else {
92.         // Caso tenha que adicionar um valor para cima
93.         xaux = mat[apr+4][0];
94.         o0aux = mat[apr+4][1];
95.         o1aux = (o0aux-o03)/(xaux-x3);
96.         o2aux = (o1aux-o12)/(xaux-x2);
97.         o3aux = (o2aux-o21)/(xaux-x1);
98.         o4aux = (o3aux-o30)/(xaux-x0);
99.     }
100.
101.     // Erro aproximado
102.     r = (pto-xaux)*(pto-x0)*(pto-x1)*(pto-x2)*(pto-x3)*o4aux;
103.
104.     // Impressão do resultado
105.     printf("\nValor: %.2f\nAproximação Solicitada:%f\nAproximação
do erro: %f\n", pto, f, r);
106.
107.     return 0;
108. }
109.
110. // Função que retorna o índice do números aproximados de pto
111. int aprox(float pto, float mat[100][2], int total) {
112.     int i;
113.     for (i=0; i<total-2; i++) {
114.         if (pto>=mat[i][0] && pto<mat[i+1][0]) {
115.             return i;
116.         }
117.     }
118.     // caso nenhum valor for encontrado entre os indices 0 e 27,
atribui-se 28.
119.     return 28;
120. }
121.
122.

```

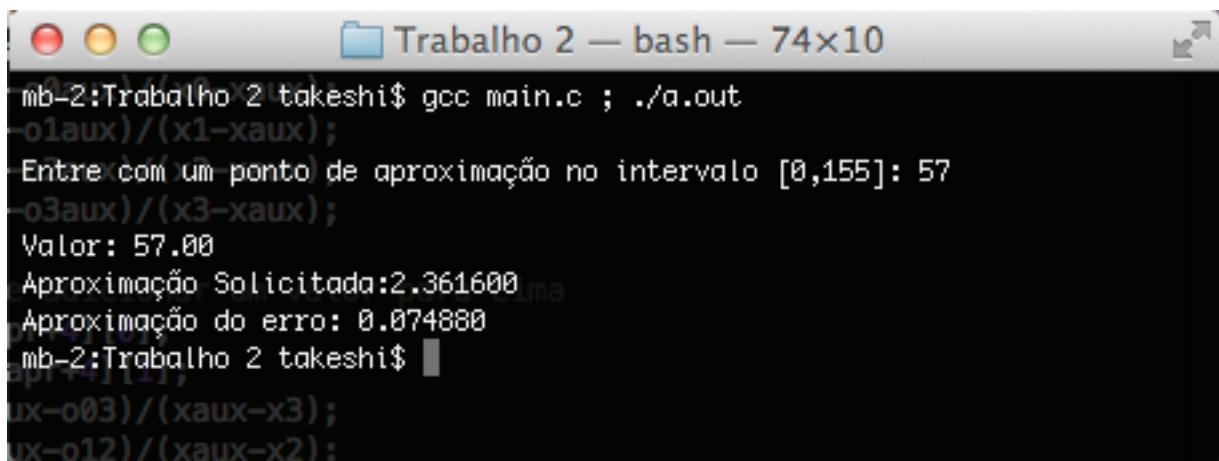
Saída do programa:

- Quando Digitado um valor fora do interval [0, 155]



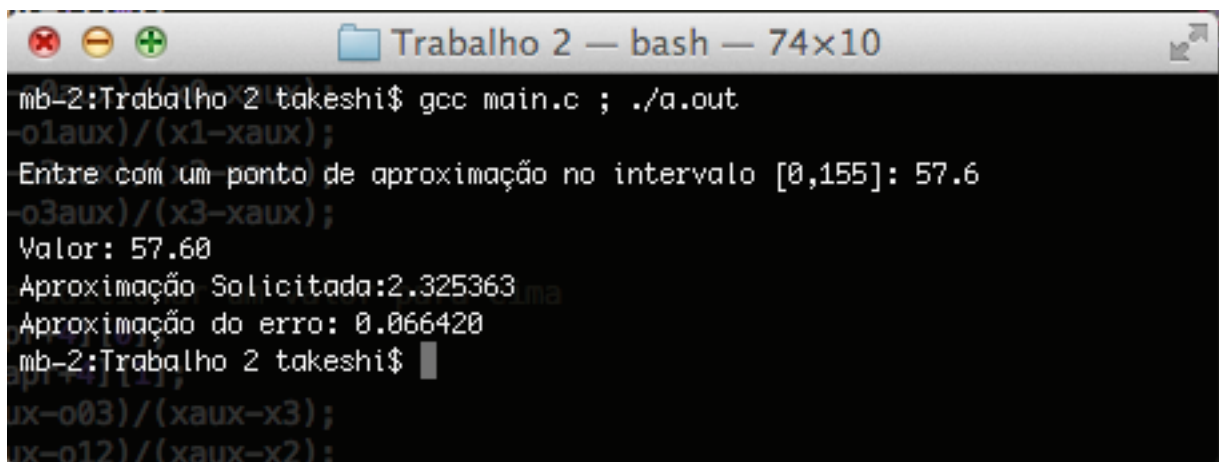
```
mb-2:Trabalho 2 takeshi$ gcc main.c ; ./a.out
o1aux)/(x1-xaux);
Entre com um ponto de aproximação no intervalo [0,155]: 156
Erro: 0 valor digitado está fora do intervalo [0, 155].
Entre com um ponto de aproximação no intervalo [0,155]:
```

- Quando entrada é o valor 57, retorna os seguintes valores:



```
mb-2:Trabalho 2 takeshi$ gcc main.c ; ./a.out
o1aux)/(x1-xaux);
Entre com um ponto de aproximação no intervalo [0,155]: 57
o3aux)/(x3-xaux);
Valor: 57.00
Aproximação Solicitada:2.361600
Aproximação do erro: 0.074880
mb-2:Trabalho 2 takeshi$
```

- Quando entrada é o valor 57,60 , retorna os seguintes valores:



```
mb-2:Trabalho 2 takeshi$ gcc main.c ; ./a.out
o1aux)/(x1-xaux);
Entre com um ponto de aproximação no intervalo [0,155]: 57.6
o3aux)/(x3-xaux);
Valor: 57.60
Aproximação Solicitada:2.325363
Aproximação do erro: 0.066420
mb-2:Trabalho 2 takeshi$
```