

PLAYAPC

PLAYAPC

Biblioteca gráfica para programadores inexperientes

Sinayra Pascoal Cotts Moreira
Universidade de Brasília

Prof. Dr. José Carlos Loureiro Ralha
Universidade de Brasília

Prof. Dr. Alexandre Zaghetto,
Universidade de Brasília



A JOHN WILEY & SONS, INC., PUBLICATION

Copyright ©2007 by John Wiley & Sons, Inc. All rights reserved.

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning, or otherwise, except as permitted under Section 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600, or on the web at www.copyright.com. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008.

Limit of Liability/Disclaimer of Warranty: While the publisher and author have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. No warranty may be created or extended by sales representatives or written sales materials. The advice and strategies contained herein may not be suitable for your situation. You should consult with a professional where appropriate. Neither the publisher nor author shall be liable for any loss of profit or any other commercial damages, including but not limited to special, incidental, consequential, or other damages.

For general information on our other products and services please contact our Customer Care Department with the U.S. at 877-762-2974, outside the U.S. at 317-572-3993 or fax 317-572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print, however, may not be available in electronic format.

Library of Congress Cataloging-in-Publication Data:

Survey Methodology / Robert M. Groves . . . [et al].
p. cm.—(Wiley series in survey methodology)
“Wiley-Interscience.”
Includes bibliographical references and index.
ISBN 0-471-48348-6 (pbk.)
1. Surveys—Methodology. 2. Social sciences—Research—Statistical methods. I. Groves, Robert M. II. Series.

HA31.2.S873 2007
001.4'33—dc22
Printed in the United States of America.

2004044064

10 9 8 7 6 5 4 3 2 1

À todos os alunos que
queiram fazer trabalhos
bonitinhos na primeira
matéria de computação
da UnB

CONTRIBUTORS

CONTENTS IN BRIEF

PART I ALGORITMOS SEQUENCIAIS, CONDICIONAIS E COM REPETIÇÕES

1 Algoritmos sequenciais	3
2 Algoritmos condicionais	9
3 Algoritmos com repetição	13

PART II ESTRUTURA DE DADOS N-DIMENSIONAIS HOMOGÊNEAS

4 Vetores	21
5 Matrizes	23

CONTEÚDO

List of Figures	xi
List of Tables	xiii
Preface	xv
Acknowledgments	xvii
Acronyms	xix
Glossary	xxi
List of Symbols	xxiii
Introduction	xxv
Sinayra Pascoal Cotts Moreira.	
References	xxvi

PART I ALGORITMOS SEQUENCIAIS, CONDICIONAIS E COM REPETIÇÕES

1 Algoritmos sequenciais	3
1.1 Resumo	3
	ix

x CONTEÚDO

1.2	Plano Cartesiano	3
1.3	Boneco Palito	4
1.4	Estrela de Davi	6
2	Algoritmos condicionais	9
2.1	Resumo	9
2.2	Quadrante de uma reta	9
3	Algoritmos com repetição	13
3.1	Resumo	13
3.2	Galáxia espiral	13
3.3	Carro andando	15
3.4	Moinho de vento	16
 PART II ESTRUTURA DE DADOS N-DIMENSIONAIS HOMOGÊNEAS		
4	Vetores	21
4.1	Resumo	21
4.2	Criando um gráfico	21
5	Matrizes	23
5.1	Resumo	23
5.2	Criando um gráfico	23

LIST OF FIGURES

1.1	Plano Cartesiano de -100 à 100	4
1.2	Boneco Palito	5
1.3	Estrela de Davi	6
2.1	Determinação do quadrante de uma reta baseado no ângulo de inclinação dela	10
3.1	Duas espirais hiperbólicas girando	14
3.2	Carro se movendo da posição -100 até a posição 100	15
3.3	Moinho de vento	16
4.1	Gráfico do polinômio $-x^3$	22
5.1	Gráfico do polinômio $-x^3$	24

LIST OF TABLES

PREFACE

This is an example preface. This is an example preface. This is an example preface. This is an example preface.

R. K. Watts

Durham, North Carolina
September, 2007

ACKNOWLEDGMENTS

From Dr. Jay Young, consultant from Silver Spring, Maryland, I received the initial push to even consider writing this book. Jay was a constant “peer reader” and very welcome advisor during this year-long process.

To all these wonderful people I owe a deep sense of gratitude especially now that this project has been completed.

G. T. S.

ACRONYMS

UnB	Universidade de Brasília
APC	Análise e Programação de Algoritmos

GLOSSARY

NormGibbs	Draw a sample from a posterior distribution of data with an unknown mean and variance using Gibbs sampling.
pNull	Test a one sided hypothesis from a numerically specified posterior CDF or from a sample from the posterior
sintegral	A numerical integration using Simpson's rule

SYMBOLS

- A Amplitude
- $\&$ Propositional logic symbol
- a Filter Coefficient
- \mathcal{B} Number of Beats

INTRODUCTION

Sinayra Pascoal Cotts Moreira.

Departamento de Ciência da Computação - UnB
Brasília, DF, Brasil

O índice de reprovação nas matérias iniciais do curso de Ciência da Computação da UnB tem crescido a cada semestre, bem como o índice de evasão. Apesar das tentativas de criar mais horários de plantão de dúvidas e maior disponibilidade dos monitores para essas disciplinas, o desinteresse se mantém. Visando aumentar o interesse dos alunos pelo curso, está sendo desenvolvida uma biblioteca gráfica 2D simplificada denominada playAPC. Para o discente, a playAPC deve ser usada para consolidar os conceitos aprendidos em Análise e Programação de Algoritmos (APC) através de modelagem gráfica. Dessa forma, os alunos podem interagir com outras disciplinas do curso de modo lúdico.

A playAPC foi desenvolvida utilizando a linguagem C++, a API OpenGL e a biblioteca GLFW 2.7. A API OpenGL deve ser suportada pela placa de vídeo presente no computador, sendo exigido a versão 1.3 no mínimo. O tutorial para instalação tanto da GLFW quanto da própria playAPC está disponível em detalhes no site Guia de Referência da playAPC ¹. Apesar da playAPC ter sido desenvolvida em C++, o seu uso é focado primariamente para alunos que estejam a programar em C, ou seja, não é necessário conhe-

¹<http://pt-br.playcb.wikia.com/wiki/Categoria:Instala%C3%A7%C3%A3o>

imento de C++ para utilizar a biblioteca, apenas utilizar a toolchain do g++ para compilar.

Neste livro, será disponibilizado uma série de exercícios usando da playAPC focando auxiliar os professores da Univerdade de Brasília (UnB) a desenvolverem novas práticas de laboratórios das turmas de APC.

REFERENCES

- [1] OpenGL SuperBible. Pearson Education Inc, 6 edition, 2014.
- [2] Marcus Geelnard and Camilla Berglund. GLFW - Reference guide, 2010. API version 2.7.
- [3] Brian W. Kernighan and Dennis M. Ritchie. The C Programming Language. 1989.
- [4] Stanley B. Lippman, Josés Lajoile, and Barbara Moo. C++ Primer. 2013.

PARTE I

ALGORITMOS SEQUENCIAIS, CONDICIONAIS E COM REPETIÇÕES

CAPÍTULO 1

ALGORITMOS SEQUENCIAIS

1.1 Resumo

Estrutura sequencial é um conjunto de instruções que serão executadas em sequência. A sequência de cada instrução deve ser seguida apara a realização de uma tarefa.

1.2 Plano Cartesiano

Esta prática se refere a exibir um Plano Cartesiano na tela com espaçamento de 5 em 5 unidades, tanto no eixo x quanto no eixo y. Com ela, o aluno poderá notar a importância da ordem de chamada de funções da playCB e a necessidade das funções AbreJanela e Desenha, além de verificar, com um exemplo simples, se a playCB foi corretamente bem instalada.

Listagem 1.1 Código fonte de Plano Cartesiano

```
1 | #include <playAPC/playapc.h>
2 | int main(){
3 |
```

playAPC, Primeira edição.
By Sinayra P.C. Moreira Copyright © 2016 John Wiley & Sons, Inc.

4 ALGORITMOS SEQUENCIAIS

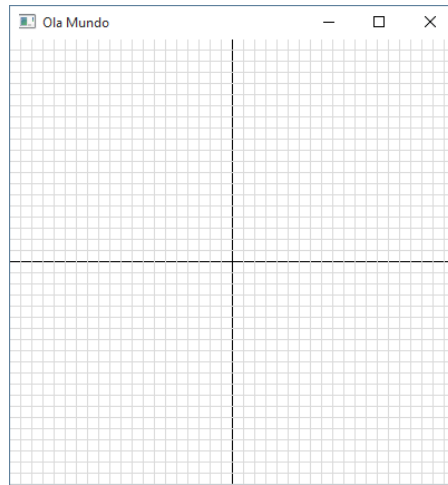


Figura 1.1 Plano Cartesiano de -100 à 100

```
4   AbreJanela(400, 400, "Ola Mundo");
5
6   PintarFundo(255, 255, 255);
7   MostraPlanoCartesiano(5);
8
9   Desenha();
10 }
```

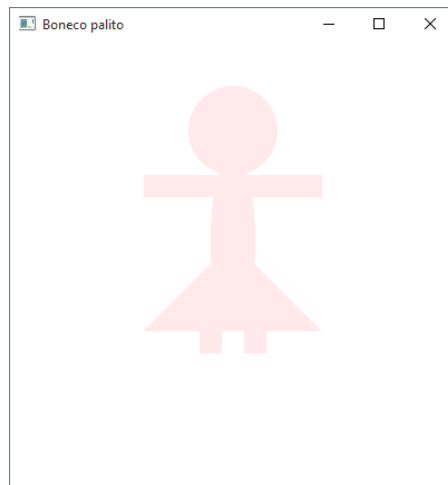
1.3 Boneco Palito

Esta prática se refere a exibir um boneco palito e praticar todas as geometrias pré-definidas existentes na playCB. Os argumentos de cada função podem ser consultados no Guia de Referência da playCB ¹

Listagem 1.2 Código fonte do boneco palito

```
1  #include <playAPC/playapc.h>
2
3  int main(){
4      Ponto p;
5      AbreJanela(400, 400, "Boneco palito");
6      PintarFundo(255, 255, 255);
7
8      p.x = 0;
9      p.y = 60;
10     CriaCirculo(20, p);
11     Pintar(255, 233, 234);
12 }
```

¹<http://pt-br.playcb.wikia.com/wiki/Categoria:Geometrias>

**Figura 1.2** Boneco Palito

```
13  p.y = 10;
14  CriaElipse(10, 40, p);
15  Pintar(255, 233, 234);
16
17  p.x = -40;
18  p.y = 30;
19  CriaRetangulo(80, 10, p);
20  Pintar(255, 233, 234);
21
22  p.x = -40;
23  p.y = -30;
24  CriaTriangulo(80, 40, p);
25  Pintar(255, 233, 234);
26
27  p.x = -15;
28  p.y = -40;
29  CriaQuadrado(10, p);
30  Pintar(255, 233, 234);
31
32  p.x = 5;
33  p.y = -40;
34  CriaQuadrado(10, p);
35  Pintar(255, 233, 234);
36
37  Desenha();
38 }
```

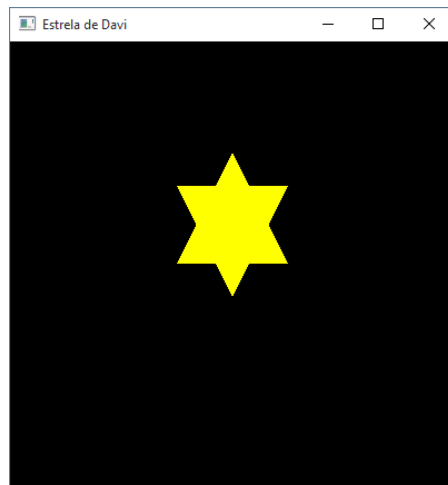



Figura 1.3 Estrela de Davi

1.4 Estrela de Davi

Esta prática se refere a exibir a estrela de Davi, feita com dois triângulos. Um triângulo foi criado com a função `CriaTriangulo` e o outro com a função `CriaPoligono`. Verificamos nesta prática os argumentos de `CriaTriangulo` (base, altura e ponto esquerdo de referência) e, como não há como ter altura negativa, teve a necessidade de criar um polígono definido pelos três pontos `p1`, `p2` e `p3` para criar-se um triângulo de cabeça pra baixo.

Listagem 1.3 Código fonte da Estrela de Davi

```

1  #include <playAPC/playapc.h>
2
3  int main(){
4      Ponto p1, p2, p3;
5      AbreJanela(400, 400, "Estrela de Davi");
6
7      p1.x = -25;
8      p1.y = 0;
9      CriaTriangulo(50, 50, p1);
10     Pintar(255, 255, 0);
11
12     p1.x = -25;
13     p1.y = 35;
14
15     p2.x = 25;
16     p2.y = 35;
17
18     p3.x = 0;
19     p3.y = -15;
20     CriaPoligono(3, p1, p2, p3);
21     Pintar(255, 255, 0);

```

```
22 ||  
23 || Desenha() ;  
24 || }
```


CAPÍTULO 2

ALGORITMOS CONDICIONAIS

2.1 Resumo

Estrutura condicional expõe que a instrução ou bloco de instrução só seja executada se a condição for verdadeira.

2.2 Quadrante de uma reta

Esta prática exibe uma reta com cor variada de acordo com qual quadrante ela pertence. A função Pintar neste caso se refere a única geometria criada no programa, no caso, a reta.

Listagem 2.1 Código fonte do quadrante da reta

```
1 | #include <playAPC/playapc.h>
2 | #include <stdio.h>
3 | #include <math.h>
4 |
5 | #define HIP 50
6 | #define PI 3.14
7 | int main() {
```

playAPC, Primeira edição.
By Sinayra P.C. Moreira Copyright © 2016 John Wiley & Sons, Inc.

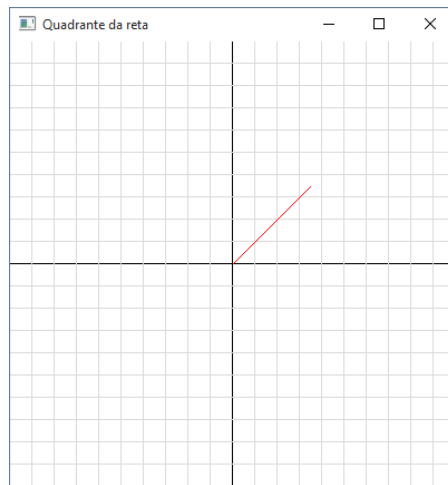


Figura 2.1 Determinação do quadrante de uma reta baseado no ângulo de inclinação dela

```

8      int angulo;
9      float anguloRad;
10     Ponto p1, p2;
11
12     printf("Digite um angulo de 0 a 360 graus:");
13     scanf("%d", &angulo);
14
15     p1.x = 0;
16     p2.x = 0;
17
18     anguloRad = (PI * angulo)/180;
19
20     p2.y = sin(anguloRad) * HIP;
21     p2.x = cos(anguloRad) * HIP;
22
23     AbreJanela(400, 400, "Quadrante da reta");
24     PintarFundo(255, 255, 255);
25     MostraPlanoCartesiano(10);
26
27     CriaReta(p1, p2);
28
29     if(p2.x > 0){
30         if(p2.y > 0)
31             Pintar(255, 0, 0); //vermelho: 1 quadrante
32         else
33             Pintar(0, 0, 0); //preto: 4 quadrante
34     }
35     else{
36         if(p2.y > 0)
37             Pintar(0, 255, 0); //verde: 2 quadrante
38         else
39             Pintar(0, 0, 255); //azul: 3 quadrante
40     }

```

```
41 ||  
42 || Desenha() ;  
43 ||  
44 || }
```


CAPÍTULO 3

ALGORITMOS COM REPETIÇÃO

3.1 Resumo

Estruturas de repetição são criadas para que diversas instruções sejam executadas um determinado número de vezes, enquanto a condição se manter verdadeira.

3.2 Galáxia espiral

Esta prática ilustra como a função `Desenha1Frame` funciona. Na linha 23 até a linha 30, a cada frame são criados dois pontos, um de cada espiral.¹

Listagem 3.1 Código fonte da galáxia expiral

```
1 | #include <playAPC/playapc.h>
2 | #include <math.h>
3 |
4 | int main (int argc, char * argv[]) {
```

¹Exemplo criado pelo Prof. Dr. José C.L. Ralha

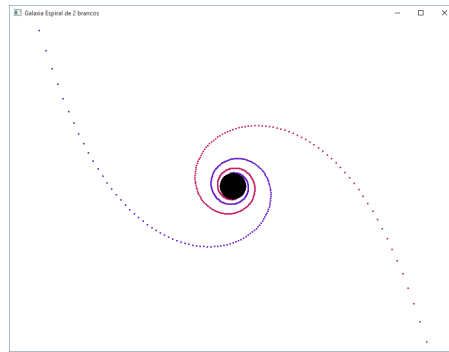


Figura 3.1 Duas espirais hiperbólicas girando

```

5
6 AbreJanela (960,960, "Galaxia Espiral de 2 brancos");
7 /*
8  Espiral Hiperbólica: equação em coordenadas cartesianas
9      x = a*cos(t)/t
10     y = a*sin(t)/t
11
12     a é a assintota para y (reta paralela ao eixo x)
13     t equivalente ao angulo em coordenadas polares
14 */
15 Ponto p, q, r;
16
17 // for (double t = 0; t < 4*PI; t += .01){
18
19     /* espiral hiperbolica , caminhando do "fim" pro centro (0,0)
20     p.x = 100*cos(t)/t;
21     p.y = 100*sin(t)/t;
22     */
23     for (double t = 4*PI; t > 0 ; t -= .05) {
24         p.x = 100*cos(t)/t;      q.x = -p.x;
25         p.y = 100*sin(t)/t;      q.y = -p.y;
26
27         CriaPonto (p);      Pintar (200, 30, 100);      Grafite(3);
28         CriaPonto (q);      Pintar (100, 30, 200);      Grafite(3);
29
30         DesenhaFrame();
31     }
32
33
34     //A massive Black Hole in the very centre
35     //If you want to see (the unseeable) black hole
36     //paint the background on a different colour
37     r.x =0;      r.y = 0;
38     CriaCirculo(8, r);
39     Pintar (0, 0, 0);
40
41     for (double t=0; ; t += .5) {
42         Gira(t);

```

```

43 Desenha1Frame();
44
45 //Depois de um tempinho, pinta o fundo de branco pra mostrar
46 //o buraco negro
47 if ( t > 200 ) PintarFundo (255, 255, 255);
48
49 //quebra o loop e encerra o programa
50 if (ApertouTecla(GLFW_KEY_ENTER)) return 0;
51 }
52
53 }

```

3.3 Carro andando

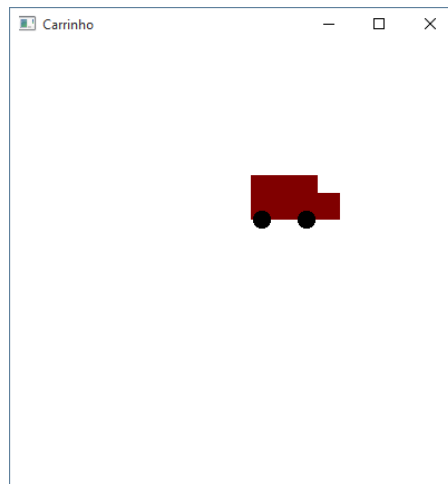


Figura 3.2 Carro se movendo da posição -100 até a posição 100

Esta prática exibe um carro construído com dois retângulos e dois círculos, agrupados com a função `CriaGrupo`, movendo-se da posição -100 até a posição 100. Nota-se que todas as geometrias que estão abaixo da função `CriaGrupo` pertencem a um único grupo, o grupo carro.

Listagem 3.2 Código fonte do carro andando

```

1 #include <playAPC/playapc.h>
2
3 int main(){
4     Ponto p;
5     int carro;
6
7     AbreJanela(400, 400, "Carrinho");
8     PintarFundo(255, 255, 255);
9

```

```

10     carro = CriaGrupo();
11     p.x = -100;
12     p.y = 20;
13     CriaRetangulo(30, 20, p);
14     Pintar(128, 0, 0);
15
16     p.x = -80;
17     p.y = 20;
18     CriaRetangulo(20, 12, p);
19     Pintar(128, 0, 0);
20
21     p.x = -95;
22     p.y = 20;
23     CriaCirculo(4, p);
24     Pintar(0, 0, 0);
25
26     p.x = -75;
27     p.y = 20;
28     CriaCirculo(4, p);
29     Pintar(0, 0, 0);
30
31     p.y = 20;
32     for (p.x = -100; p.x < 100; p.x++){
33         Move(p, carro);
34         Desenha1Frame();
35     }
36     Desenha();
37 }

```

3.4 Moinho de vento

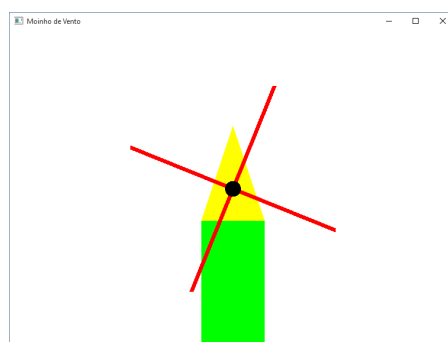


Figura 3.3 Moinho de vento

Esta prática exibe um moinho de vento criado com um grupo composto por um triângulo e um retângulo, o grupo moinho, e outro grupo composto pelas hélices, o grupo grupo. Somente o grupo sofre a ação de girar.²

Listagem 3.3 Código fonte do moinho

```

1  /* *
2  UnB - Professor : Ralha - Computação Básica
3  Aluno : Pedro Paulo de Pinho Matos
4  Matrícula : 14/0070354
5  */
6
7  #include <stdlib.h>
8  #include <stdio.h>
9  #include <playAPC/playapc.h>
10
11
12  int main (int argc, char* argv[]) {
13      int angulo =1;
14
15      AbreJanela(1024, 768, "Moinho de Vento" );
16      PintarFundo(255,255,255);
17      Ponto p,q, r;
18
19      int moinho = CriaGrupo();
20      Ponto x, y;
21      x.x = -20; x.y = -20; CriaTriangulo(40,60,x); Pintar(255,255,0);
22      y.x = -20; y.y = -100; CriaRetangulo(40,80,y); Pintar(0,255,0);
23
24      int grupo = CriaGrupo(); // Hélice 1
25      q.x = 0; q.y = 0; r.x = 0; r.y = 70;
26      CriaReta(q,r); Pintar(255,0,0); Grafite(8);
27
28      // Hélice 2
29      q.x = 0; q.y = 0; r.x = 70; r.y = 0;
30      CriaReta(q,r); Pintar(255,0,0); Grafite(8);
31
32      // Helice 3
33      q.x = 0; q.y = 0; r.x = 0; r.y = -70;
34      CriaReta(q,r); Pintar(255,0,0); Grafite(8);
35
36      // Helice 4
37      q.x = 0; q.y = 0; r.x = -70; r.y = 0;
38      CriaReta(q,r); Pintar(255,0,0); Grafite(8);
39
40      p.x = 0; p.y = 0;
41      CriaCirculo(5,p); Pintar(0,0,0); Grafite(25);
42
43      while( angulo > 0){
44          Desenha1Frame();
45          Gira(angulo,grupo);
46          angulo ++;
47      }

```

²Exemplo criado pelo aluno Pedro Paulo de Pinho Matos, da turma de Computação Básica de 1/2014

18 ALGORITMOS COM REPETIÇÃO

```
48 ||  
49 || return 0;  
50 || }
```

PARTE II

ESTRUTURA DE DADOS N-DIMENSIONAIS HOMOGÊNEAS

CAPÍTULO 4

VETORES

4.1 Resumo

Vetores são um tipo de estrutura que podem armazenar um tamanho fixo de elementos do mesmo tamanho e mesmo tipo, alocados em memória contígua. Utiliza-se vetores como um tipo de lista unidimensional, acessada através de índices.

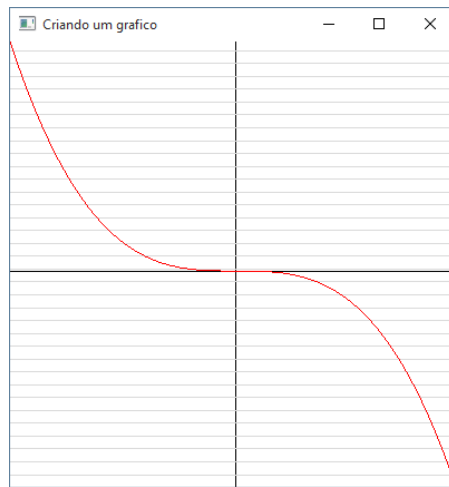
4.2 Criando um gráfico

Esta prática mostra como construir um gráfico a partir de um vetor de Pontos. Cada posição em y de cada ponto é calculada dentro do loop. Por padrão, os limites da janela de exibição da playCB vão de -100 à 100, entretanto, os valores em y nesta função variam de -125.000 até 125.000 , tendo a necessidade de mudar o limite de exibição com a função `MudaLimitesJanela(125000)`.

Listagem 4.1 Código fonte de polinômio

```
1 || #include <playAPC/playapc.h>
2 || #include <math.h>
```

playAPC, Primeira edição.
By Sinayra P.C. Moreira Copyright © 2016 John Wiley & Sons, Inc.

**Figura 4.1** Gráfico do polinômio $-x^3$

```
3
4  int main(){
5      Ponto p[100];
6      int i, j;
7
8      MudaLimitesJanela(125000);
9
10     AbreJanela(400, 400, "Criando um grafico");
11     PintarFundo(255, 255, 255);
12     MostraPlanoCartesiano(7000);
13
14     j = -50;
15     for(i = 0; i < 100; i++, j++){
16         p[i].x = j;
17         p[i].y = -pow(p[i].x, 3);
18     }
19
20     CriaGrafico(100, p, 1);
21     Pintar(255, 0, 0);
22
23     Desenha();
24
25 }
```

CAPÍTULO 5

MATRIZES

5.1 Resumo

Assim como vetores, matrizes são um tipo de estrutura que armazena dados de mesmo tamanho e mesmo tipo, mas são utilizadas de maneira n-dimensional. O modo mais comum de utilizar matriz é usando-a na forma bidimensional, onde os dados são tratados como se estivessem numa tabela, com linhas e colunas.

5.2 Criando um gráfico

Esta prática mostra como construir um gráfico a partir de um vetor de Pontos. Cada posição em y de cada ponto é calculada dentro do loop. Por padrão, os limites da janela de exibição da playCB vão de -100 à 100, entretanto, os valores em y nesta função variam de -125.000 até 125.000, tendo a necessidade de mudar o limite de exibição com a função MudaLimitesJanela(125000).

Listagem 5.1 Código fonte de polinômio

```
1 || #include <playAPC/playapc.h>
```

playAPC, Primeira edição.
By Sinayra P.C. Moreira Copyright © 2016 John Wiley & Sons, Inc.

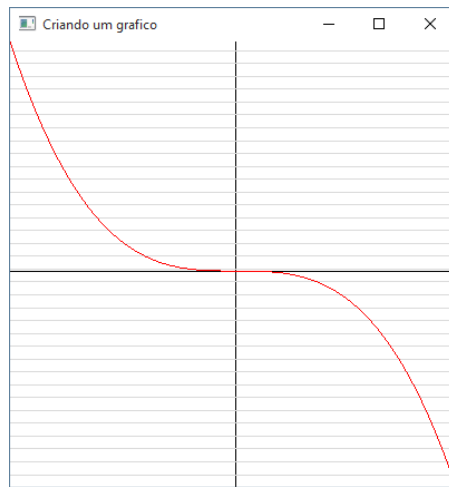


Figura 5.1 Gráfico do polinômio $-x^3$

```

2  #include <math.h>
3
4  int main(){
5      Ponto p[100];
6      int i, j;
7
8      MudaLimitesJanela(125000);
9
10     AbreJanela(400, 400, "Criando um grafico");
11     PintarFundo(255, 255, 255);
12     MostraPlanoCartesiano(7000);
13
14     j = -50;
15     for(i = 0; i < 100; i++, j++){
16         p[i].x = j;
17         p[i].y = -pow(p[i].x, 3);
18     }
19
20     CriaGrafico(100, p, 1);
21     Pintar(255, 0, 0);
22
23     Desenha();
24
25 }
```