Desenho com Primitivas

Generated by Doxygen 1.8.17

Chapter 1

projeto-desenho

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 comando_t Struct Reference

Struct de comando.

`#include <structs.h>`

Public Attributes

- char nome_comando [20]
- int qtd_parametros
- int parametros [30]
- char comando_string [30]

4.1.1 Detailed Description

Struct de comando.

Recebe as instruções de cada linha do arquivo de input.

Definition at line 6 of file structs.h.

4.1.2 Member Data Documentation

4.1.2.1 comando_string

`char comando_t::comando_string[30]`

Parêmetros em String da primitiva

Definition at line 11 of file structs.h.

4.1.2.2 nome_comando

```
char comando_t::nome_comando[20]
```

Nome da primitiva.

Definition at line 8 of file structs.h.

4.1.2.3 parametros

```
int comando_t::parametros[30]
```

Parâmetros da primitiva.

Definition at line 10 of file structs.h.

4.1.2.4 qtd_parametros

```
int comando_t::qtd_parametros
```

Quantidade de parâmetros passados na primitiva.

Definition at line 9 of file structs.h.

The documentation for this struct was generated from the following file:

- structs.h

4.2 imagem_t Struct Reference

Struct da Imagem.

```
#include <structs.h>
```

Public Attributes

- int *** matriz
- int altura
- int largura
- int cor_atual [3]
- char nome_imagem [31]

4.2.1 Detailed Description

Struct da Imagem.

Armazena os pixels da imagem, além de dados relacionados a ela tais como as dimensões, nome e a cor usada na primitiva atual.

Definition at line 18 of file structs.h.

4.2.2 Member Data Documentation

4.2.2.1 altura

`int imagem_t::altura`

Altura da imagem.

Definition at line 21 of file structs.h.

4.2.2.2 cor_atual

`int imagem_t::cor_atual[3]`

Cor atual para uso nas primitivas.

Definition at line 23 of file structs.h.

4.2.2.3 largura

`int imagem_t::largura`

Largura da imagem.

Definition at line 22 of file structs.h.

4.2.2.4 matriz

`int*** imagem_t::matriz`

Vetor tridimensional de pixels.

Definition at line 20 of file structs.h.

4.2.2.5 nome_imagem

`char imagem_t::nome_imagem[31]`

Nome do arquivo de imagem.

Definition at line 24 of file structs.h.

The documentation for this struct was generated from the following file:

- structs.h

Chapter 5

File Documentation

5.1 funcoes.h File Reference

Functions

- void split (char string_entrada[ ], char delim[ ], char retorno[ ][50])
  Reparte uma string.
- void limpar_string_arquivo (char entrada[ ], char saida[ ])
  Limpa strings.
- void retira_barraN (char entrada[ ], char saida[ ])
  Retira a quebra de linha de uma string.
- imagem alocar_matriz ()
  Aloca a matriz da imagem.
- void realocar_matriz (imagem *img)
  Realoca a matriz de pixels com o tamanho necessário.
- void desalocar_matriz (imagem *img)
  Desaloca a matriz de pixels.

5.1.1 Function Documentation

5.1.1.1 alocar_matriz()

```
imagem alocar_matriz ( )
```

Aloca a matriz da imagem.

Faz a alocação inicial do vetor tridimensional que guarda os pixels referentes a imagem que está sendo editada.

Returns

Uma instancia da struct imagem com a matriz tridimensional alocada.

Definition at line 118 of file funcoes.c.

```
118                         {
119
120     imagem img;
121
122     img.matriz = malloc(sizeof(int**) * 1);
123     img.matriz[0] = malloc(sizeof(int*) * 1);
124     img.matriz[0][0] = malloc(sizeof(int) * 3);
125
126     return img;
127 }
```

5.1.1.2 desalocar_matriz()

```
void desalocar_matriz (
            imagem * img )
```

Desaloca a matriz de pixels.

Usa a informação recebida da struct imagem, para pegar a resolução da imagem a ser editada para desalocar todos os bytes que foram alocados na matriz de acordo com essa resolução.

Parameters

img	recebe o ponteiro com a imagem a ser desalocada.

Definition at line 98 of file funcoes.c.

```
98                                      {
99
100     //puts("desalocando a matriz");
101     for (int i = 0; i < img->altura; ++i)
102     {
103         for (int j = 0; j < img->largura; ++j)
104         {
105             free(img->matriz[i][j]);
106
107         }
108     }
109
110     for (int i = 0; i < img->altura; ++i)
111     {
112         free(img->matriz[i]);
113     }
114
115     free(img->matriz);
116 }
```

5.1.1.3 limpar_string_arquivo()

```
void limpar_string_arquivo (
             char entrada[],
             char saida[] )
```

Limpa strings.

Retira caracteres especiais e outras "sujeiras" que possam estar na string remanescentes daquela posição da memória

Parameters

entrada	recebe a string a ser limpa
saida	recebe uma string para depositar a string após ser limpa

Definition at line 79 of file funcoes.c.

```
79                                                      {
80
81     int i = 0;
82     while(1){
83
84         if(isalpha(entrada[i]) || entrada[i] == '.' || entrada[i] == '_' || numero(entrada[i])){
85
86             saida[i] = entrada[i];
```

```
87
88          } else if(entrada[i] == '\0'){
89
90              break;
91          }
92
93          i++;
94      }
95 }
```

5.1.1.4 realocar_matriz()

```
void realocar_matriz (
              imagem * img )
```

Realoca a matriz de pixels com o tamanho necessário.

Usa a informação recebida da struct imagem, para pegar a resolução da imagem a ser editada para realocar a matriz com a quantidade de bytes de bytes necessária para armazenar todos os pixels da imagem com tal resolução.

Parameters

img	recebe o ponteiro com a imagem a ser realocada.

Definition at line 130 of file funcoes.c.

```
130                                 {
131
132     //printf("Alt: %d\n", img->altura);
133     //printf("Larg: %d\n", img->largura);
134     //puts("realocando...");
135     img->matriz = realloc(img->matriz, sizeof(int**) * img->altura);
136
137     for (int i = 0; i < img->altura; ++i)
138     {
139         img->matriz[i] = realloc(img->matriz[i], sizeof(int*) * img->largura);
140     }
141
142     for (int i = 0; i < img->altura; ++i)
143     {
144         for (int j = 0; j < img->largura; ++j)
145         {
146             img->matriz[i][j] = realloc(img->matriz[i][j], sizeof(int) * 3);
147         }
148     }
149     //puts("realocou");
150 }
```

5.1.1.5 retira_barraN()

```
void retira_barraN (
              char entrada[],
              char saida[] )
```

Retira a quebra de linha de uma string.

Parameters

entrada	recebe a string para ser retirada a quebra de linha.
saida	recebe a string para ser colocada a string limpa.

5.1.1.6 split()

```
void split (
            char string_entrada[],
            char delim[],
            char retorno[][50] )
```

Reparte uma string.

Recebe uma string e quebra ela em um vetor de strings de acordo com uma string usada como separadora.

Parameters

string_entrada	recebe a string a ser repartida.
delim	recebe a string utilizada para quebrar a string em substrings.
retorno	recebe um vetor de string que é usado para colocar a substrings geradas

Definition at line 8 of file funcoes.c.

```
8                                                                                   {
9
10      int tam_delimitador = 0;
11      int indice_str_atual = 0;
12      int indice_retorno = 0;
13
14      while(delim[tam_delimitador] != '\0'){
15          tam_delimitador++;
16      }
17
18      int i = 0;
19      while(string_entrada[i] != '\0'){
20
21          if(string_entrada[i] == delim[0]){
22
23              int corresponde = 1;
24              int loops = 0;
25              for (int j = 1; j < tam_delimitador; ++j)
26              {
27                  loops++;
28
29                  if(string_entrada[i + j] != delim[j] ){
30
31                      corresponde = 0;
32                      break;
33                  }
34              }
35
36              if(corresponde == 1){
37
38                  retorno[indice_retorno][indice_str_atual] = '\0';
39                  i += tam_delimitador -1;
40                  indice_retorno++;
41                  indice_str_atual = 0;
42
43              } else {
44
45                  for (int l = i; l <= i + loops; ++l)
46                  {
47                      retorno[indice_retorno][indice_str_atual] = string_entrada[l];
48                      indice_str_atual++;
49                  }
50                  i += loops;
51              }
52
53          } else {
54
55              retorno[indice_retorno][indice_str_atual] = string_entrada[i];
56              indice_str_atual++;
57          }
58          i++;
59      }
```

```
60      retorno[indice_retorno][indice_str_atual] = '\0';
61  }
```

5.2 primitivas.h File Reference

Functions

- void save (imagem *img, char nome_arquivo[ ])
  Salva a Imagem.
- void line (imagem *img, int parametros[ ])
  Cria uma linha.
- void polygon (imagem *img, comando cmd)
  Cria um polígono.
- void circle (imagem *img, int parametros[ ])
  Cria um círculo.
- void fill (imagem *img, int parametros[ ])
  Preenche superfícies.
- void color (imagem *img, int parametros[ ])
  Define uma cor.
- void clear (imagem *img, int parametros[ ])
  Limpa uma imagem.
- void rect (imagem *img, int parametros[ ])
  Cria um retângulo.
- void repeat_line (imagem *img, int parametros[ ], comando *ultima_entrada)
  Repete uma linha.
- void repeat_polygon (imagem *img, int parametros[ ], comando *ultima_entrada)
  Repete um polígono.
- void copy_polygon (imagem *img, int parametros[ ], comando *ultima_entrada)
  Copia um polígono.
- void open (imagem *img, char nome_arquivo[ ])
  Abre uma imagem.
- void image (imagem *img, int parametros[ ])
  Define a resolução da imagem a ser criada.

5.2.1 Function Documentation

5.2.1.1 circle()

```
void circle (
            imagem * img,
            int parametros[ ] )
```

Cria um círculo.

A função cria um círculo recebendo as coordenadas do ponto central e o raio.

Parameters

img	recebe o ponteiro com a imagem a salva.
cmd	recebe uma struct comando, de onde é retirado os parâmetros necessários para a execução da matriz.

Definition at line 242 of file primitivas.c.

```
242                                             {
243
244     int raio = parametros[2];
245     int x0 = parametros[0];
246     int y0 = parametros[1];
247     int f = 1 - raio;
248     int ddF_x = 0;
249     int ddF_y = -2 * raio;
250     int x = 0;
251     int y = raio;
252
253     for (int i = 0; i < 3; ++i)
254     {
255         img->matriz[x0][y0 + raio][i] = img->cor_atual[i];
256         img->matriz[x0][y0 - raio][i] = img->cor_atual[i];
257         img->matriz[x0 + raio][y0][i] = img->cor_atual[i];
258         img->matriz[x0 - raio][y0][i] = img->cor_atual[i];
259     }
260
261     while(x < y){
262
263         if(f >= 0){
264
265             y--;
266             ddF_y += 2;
267             f += ddF_y;
268         }
269
270         x++;
271         ddF_x += 2;
272         f += ddF_x + 1;
273
274         for (int i = 0; i < 3; ++i)
275         {
276             img->matriz[x0 + x][y0 + y][i] = img->cor_atual[i];
277             img->matriz[x0 - x][y0 + y][i] = img->cor_atual[i];
278             img->matriz[x0 + x][y0 - y][i] = img->cor_atual[i];
279             img->matriz[x0 - x][y0 - y][i] = img->cor_atual[i];
280             img->matriz[x0 + y][y0 + x][i] = img->cor_atual[i];
281             img->matriz[x0 - y][y0 + x][i] = img->cor_atual[i];
282             img->matriz[x0 + y][y0 - x][i] = img->cor_atual[i];
283             img->matriz[x0 - y][y0 - x][i] = img->cor_atual[i];
284         }
285     }
286 }
```

5.2.1.2 clear()

```
void clear (
              imagem * img,
              int parametros[] )
```

Limpa uma imagem.

A função recebe uma cor para preencher todos os pixels da imagem.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor contendo os três valores (rgb) para definir a cor a ser usada no preenchimento.

Definition at line 208 of file primitivas.c.

```
208                                             {
209
210     for(int i = 0; i < img->altura; i++)
211     {
212         for (int j = 0; j < img->largura; ++j)
213         {
214             img->matriz[i][j][0] = parametros[0];
215             img->matriz[i][j][1] = parametros[1];
216             img->matriz[i][j][2] = parametros[2];
217         }
218     }
219 }
```

5.2.1.3 color()

```
void color (
              imagem * img,
              int parametros[] )
```

Define uma cor.

A função define uma cor (rgb) a ser usada para as primitivas que se seguem a essa instrução.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor contendo os três valores (rgb) para definir a cor.

Definition at line 199 of file primitivas.c.

```
199                                             {
200
201     for (int i = 0; i < 3; ++i)
202     {
203         img->cor_atual[i] = parametros[i];
204     }
205 }
```

5.2.1.4 copy_polygon()

```
void copy_polygon (
              imagem * img,
              int parametros[],
              comando * ultima_entrada )
```

Copia um polígono.

Define uma nova coordenada e um novo polígono é copiado para essa nova posição, tomando como referência seu primeiro ponto.

O polígono a ser copiado é o último definido.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor contendo a quantidade de repetições e os incrementos.
ultima_entrada	recebe um ponteiro referente a uma struct comando que contem a última instrução que contem o polígono a ser repetido.

Definition at line 182 of file primitivas.c.

```
182                                                                                 {
183
184     int distanciax = parametros[0] - ultima_entrada->parametros[1];
185     int distanciay = parametros[1] - ultima_entrada->parametros[2];
186     int pontos = ultima_entrada->parametros[0];
187     int qtd_parametros = ultima_entrada->qtd_parametros;
188
189     for (int i = 0; i < pontos * 2; i += 2)
190     {
191         ultima_entrada->parametros[i + 1] += distanciax;
192         ultima_entrada->parametros[i + 2] += distanciay;
193     }
194
195     polygon(img, *ultima_entrada);
196 }
```

5.2.1.5 fill()

```
void fill (
              imagem * img,
              int parametros[] )
```

Preenche superfícies.

A função preenche figuras ou qualquer superfície da imagem com a cor atual, até encontrar uma borda.

Para isso a função recebe as coodernadas do ponto onde será pintado.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor, sendo as primeiras posições a coordenadas do ponto central, seguido do raio.

Definition at line 332 of file primitivas.c.

```
332                                         {
333
334     int x = parametros[0];
335     int y = parametros[1];
336     int cor_inicial[3];
337     int pintado = 1;
338
339     for (int i = 0; i < 3; ++i)
340     {
341         cor_inicial[i] = img->matriz[y][x][i];
342     }
343
344     for (int i = 0; i < 3; ++i)
345     {
346        if(img->cor_atual[i] != img->matriz[y][x][i]){
347
348             pintado = 0;
349             break;
350         }
351     }
352
353     if(pintado == 0){
354
355         rec_fill(img, x, y, cor_inicial);
356     }
357 }
```

5.2.1.6 image()

```
void image (
```

```
            imagem * img,
            int parametros[] )
```

Define a resolução da imagem a ser criada.

A função cria uma imagem a se definir a resolução.

A função recebe no vetor de parâmetro os dois valores referentes a quantidade de pixels no eixo x e y, chamando a função de alocação em seguida, reservando a memória necessária para a imagem.

Parameters

img	recebe o ponteiro com a imagem e salva a resolução nela, além de passá-la para fazer a alocação.
parametros	recebe um vetor com os valores referentes a resolução da imagem a ser criada.

Definition at line 429 of file primitivas.c.

```
429                                             {
430
431     img->altura = parametros[1];
432     img->largura = parametros[0];
433     //printf("Largura: %d\n", img->largura);
434     //printf("Altura: %d\n", img->altura);
435     realocar_matriz(img);
436 }
```

5.2.1.7 line()

```
void line (
            imagem * img,
            int parametros[] )
```

Cria uma linha.

A função cria uma linha entre duas coordenadas x y dadas.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor com o ponto inicial e final da reta.

Definition at line 30 of file primitivas.c.

```
30                                          {
31
32     int x0 = parametros[0];
33     int x1 = parametros[2];
34     int y0 = parametros[1];
35     int y1 = parametros[3];
36     int err, dy, dx, sx, sy, e2;
37
38     dx = abs(x1 - x0);
39
40     if(x0 < x1){
41
42         sx = 1;
43
44     } else {
45
46         sx = -1;
47     }
48
```

```
49      dy = abs(y1 - y0) * -1;
50
51      if(y0 < y1){
52
53          sy = 1;
54
55      } else {
56
57          sy = -1;
58      }
59
60      err = dx+dy;
61      while (1){
62
63          for (int i = 0; i < 3; ++i)
64          {
65             img->matriz[y0][x0][i] = img->cor_atual[i];
66          }
67
68          if (x0==x1 && y0==y1){
69              break;
70          }
71
72          e2 = 2*err;
73
74          if (e2 >= dy){
75
76              err += dy;
77              x0 += sx;
78          }
79
80          if (e2 <= dx){
81              err += dx;
82              y0 += sy;
83          }
84      }
85  }
```

5.2.1.8 open()

```
void open (
              imagem * img,
              char nome_arquivo[] )
```

Abre uma imagem.

A função lê um arquivo de imagem e copia os pixels para uma matrix de pixels, podendo ser editada pelo programa.

Parameters

img	recebe o ponteiro com a imagem para qual a informação vinda do arquivo de imagem será salvada.
nome_arquivo	recebe o nome do arquivo de imagem que será aberto.

Definition at line 360 of file primitivas.c.

```
360                                             {
361
362      char text[15];
363      char text_split[3][50];
364      char altura[5];
365      char largura[5];
366      char nome_arquivo_tratado[30];
367      int qtd_cores;
368
369      for (int i = 0; i < 30; ++i)
370      {
371          nome_arquivo_tratado[i] = '\0';
372      }
373
374      limpar_string_arquivo(nome_arquivo, nome_arquivo_tratado);
```

```
375
376     FILE *file;
377     file = fopen(nome_arquivo_tratado, "r");
378
379     if(file == NULL){
380
381         puts("Arquivo não encontrado");
382
383     } else {
384
385         fgets(text, 10, file);
386         fgets(text, 10, file);
387
388         split(text, " ", text_split);
389
390         strcpy(altura, text_split[1]);
391         strcpy(largura, text_split[0]);
392
393         strcpy(img->nome_imagem, nome_arquivo_tratado);
394
395         sscanf(altura, "%d", &img->altura);
396         sscanf(largura, "%d", &img->largura);
397
398         realocar_matriz(img);
399
400         //printf(">>Altura: %d\n", img->altura);
401         //printf(">>Largura: %d\n", img->largura);
402
403         qtd_cores = img->altura * img->largura * 3;
404
405         char cores[(qtd_cores + 1) * 12];
406         char vetor_cores[qtd_cores][50];
407
408         fgets(text, 15, file);
409         fgets(cores, qtd_cores * 4 , file);
410
411         split(cores, " ", vetor_cores);
412
413         int indice = 0;
414         for (int i = 0; i < img->altura; ++i)
415         {
416             for (int j = 0; j < img->largura; ++j)
417             {
418                 for (int k = 0; k < 3; ++k)
419                 {
420                     sscanf(vetor_cores[indice], "%d", &img->matriz[i][j][k]);
421                     indice++;
422                 }
423             }
424         }
425     }
426 }
```

5.2.1.9 polygon()

```
void polygon (
              imagem * img,
              comando cmd )
```

Cria um polígono.

A função cria um polígono que passa pelos pontos dados.

A função recebe na primeira posição do vetor de parâmetros a quantidade de pontos. Os outros parâmetros a seguir são cada uma das ordenadas.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor com a quantidade de pontos e as coordenadas deles.

Definition at line 111 of file primitivas.c.

```
111                                     {
112 
113     int qtd_pontos = cmd.parametros[0];
114     int ordenadas = cmd.qtd_parametros -1;
115     int parametros[4];
116     int retas[ordenadas][4];
117     int indice_retas = 0;
118 
119     //cria as retas
120     for (int i = 0; i < ordenadas; i += 2)
121     {
122         if(i == ordenadas - 2) {
123 
124             retas[indice_retas][0] = cmd.parametros[i + 1];
125             retas[indice_retas][1] = cmd.parametros[i + 2];
126             retas[indice_retas][2] = cmd.parametros[1];
127             retas[indice_retas][3] = cmd.parametros[2];
128 
129             indice_retas++;
130 
131         } else {
132 
133             retas[indice_retas][0] = cmd.parametros[i + 1];
134             retas[indice_retas][1] = cmd.parametros[i + 2];
135             retas[indice_retas][2] = cmd.parametros[i + 3];
136             retas[indice_retas][3] = cmd.parametros[i + 4];
137 
138             indice_retas++;
139         }
140     }
141 
142     //coloca as retas na imagem.
143     for (int i = 0; i < indice_retas; ++i)
144     {
145         for (int j = 0; j < 4; ++j)
146         {
147 
148             parametros[j] = retas[i][j];
149 
150         }
151 
152         line(img, parametros);
153     }
154 }
```

5.2.1.10 rect()

```
void rect (
              imagem * img,
              int parametros[] )
```

Cria um retângulo.

A função cria um retângulo, recebendo o ponto inicial e o tamanho nos eixos x e y.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor contendo a coordenada e os tamanhos x e y.

Definition at line 222 of file primitivas.c.

```
222                                     {
223 
224     comando cmd;
225     cmd.qtd_parametros = 9;
226 
227     cmd.parametros[0] = 4;
228     cmd.parametros[1] = parametros[0];
229     cmd.parametros[2] = parametros[1];
```

```
230     cmd.parametros[3] = parametros[0] + parametros[2];
231     cmd.parametros[4] = parametros[1];
232     cmd.parametros[5] = parametros[0] + parametros[2];
233     cmd.parametros[6] = parametros[1] + parametros[3];
234     cmd.parametros[7] = parametros[0];
235     cmd.parametros[8] = parametros[1] + parametros[3];
236
237     polygon(img, cmd);
238 }
```

5.2.1.11 repeat_line()

```
void repeat_line (
            imagem * img,
            int parametros[],
            comando * ultima_entrada )
```

Repete uma linha.

A função repete uma linha definido a quantidade de repetições e os incrementos no x e no y. Dessa forma, a reta é repetida pela quantidade definida, somando a altura e a largura dos incrementos para gerar as novas retas.

A linha a ser repetida é a última definida.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor contendo a quantidade de repetições e os incrementos.

Definition at line 87 of file primitivas.c.

```
87                                                                  {
88
89     if(strcmp(ultima_entrada->nome_comando, "line") == 0){
90
91         int quantidade = parametros[0];
92         int deltax0 = parametros[1];
93         int deltay0 = parametros[2];
94         int deltax1 = parametros[3];
95         int deltay1 = parametros[4];
96         int novos_parametros[4];
97
98         for (int i = 0; i < quantidade; ++i)
99         {
100             novos_parametros[0] = ultima_entrada->parametros[0] + deltax0 * (i + 1);
101             novos_parametros[1] = ultima_entrada->parametros[1] + deltay0 * (i + 1);
102             novos_parametros[2] = ultima_entrada->parametros[2] + deltax1 * (i + 1);
103             novos_parametros[3] = ultima_entrada->parametros[3] + deltay1 * (i + 1);
104             line(img, novos_parametros);
105          }
106
107     }
108 }
```

5.2.1.12 repeat_polygon()

```
void repeat_polygon (
            imagem * img,
            int parametros[],
            comando * ultima_entrada )
```

Repete um polígono.

A função repete um polígono definido a quantidade de repetições e os incrementos no x e no y. Dessa forma, o polígono é repetido pela quantidade definida, somando a altura e a largura dos incrementos em cada um dos pontos, para gerar os novos polígonos.

O polígono a ser repetído é o último definido.

Parameters

img	recebe o ponteiro com a imagem a salva.
parametros	recebe um vetor contendo a quantidade de repetições e os incrementos.
ultima_entrada	recebe um ponteiro referente a uma struct comando que contem a última instrução que contem o polígono a ser repetido.

Definition at line 157 of file primitivas.c.

```
157                                                                                 {
158
159     if(strcmp(ultima_entrada->nome_comando, "polygon") == 0){
160
161         int quantidade = parametros[0];
162         int deltax = parametros[1];
163         int deltay = parametros[2];
164         int ordenadas = ultima_entrada->parametros[0] * 2;
165
166         comando novo_poligono = *ultima_entrada;
167
168         for (int i = 0; i < quantidade; ++i)
169         {
170             for (int j = 1; j < ordenadas + 1; j+=2)
171             {
172                 novo_poligono.parametros[j] += deltax;
173                 novo_poligono.parametros[j + 1] += deltay;
174             }
175
176             polygon(img, novo_poligono);
177         }
178     }
179 }
```

5.2.1.13 save()

```
void save (
              imagem * img,
              char nome_arquivo[] )
```

Salva a Imagem.

Copiar os pixels da matriz de pixel para um arquivo .ppm, gerando a imagem.

Parameters

img	recebe o ponteiro com a imagem a salva.
nome_arquivo	é passado o nome do arquivo de imagem a ser salvo.

Definition at line 9 of file primitivas.c.

```
9                                       {
10
11     FILE *file = fopen(nome_arquivo, "wb");
12     fprintf(file, "P3\n%d %d\n255\n", img->largura, img->altura);
```

```
13
14      for (int i = 0; i < img->altura; ++i)
15      {
16          for (int j = 0; j < img->largura; ++j)
17          {
18              for (int z = 0; z < 3; ++z)
19              {
20                  fprintf(file, "%d ", img->matriz[i][j][z]);
21              }
22          }
23      }
24
25      fclose(file);
26  }
```

5.3 programa.h File Reference

Functions

- void executar (char nome_arquivo[ ])

  Função Executar, chamada no main.
- void interpretar (comando *entrada, comando *ultima_entrada, imagem *img)

  Função de interpretação dos comandos.

5.3.1 Function Documentation

5.3.1.1 executar()

```
void executar (
              char nome_arquivo[ ] )
```

Função Executar, chamada no main.

Parameters

nome_arquivo	recebe uma string correspondente ao nome do arquivo de instruções.

Definition at line 82 of file programa.c.

```
82                                  {
83
84      imagem img = alocar_matriz();
85      comando instrucao;
86      comando ultima_instrucao;
87
88      strcpy(ultima_instrucao.nome_comando, "vazio");
89
90      strcpy(img.nome_imagem, "imagem.ppm");
91
92      FILE *file;
93      file = fopen(nome_arquivo, "r");
94
95      char text[50];
96      char entradas[50][50];
97      char nome_comando[50];
98      int parametros[30];
99      int qtd_parametros = 0;
100
101      //limpa os vetores
```

```
102     for (int i = 0; i < 30; ++i)
103     {
104         parametros[i] = -1;
105     }
106 
107     for (int i = 0; i < 49; ++i)
108     {
109         text[i] = ' ';
110         strcpy(entradas[i], " ");
111     }
112     text[49] = ' ';
113     strcpy(entradas[49], " ");
114 
115 
116     if(file == NULL){
117 
118         printf("Arquivo não encontrado\n");
119 
120     } else {
121 
122         while(fgets(text, 50, file) != NULL){
123 
124             qtd_parametros = 0;
125             split(text, " ", entradas);
126 
127             strcpy(nome_comando, entradas[0]);
128 
129             strcpy(instrucao.nome_comando, nome_comando);
130 
131             if(strcmp(nome_comando, "open") == 0 || strcmp(nome_comando, "save") == 0 ){
132 
133                 strcpy(instrucao.comando_string, entradas[1]);
134             }
135 
136             // separando os comandos
137             for (int i = 0; i < 29; ++i)
138             {
139                 //converte a string vindo de entradas para o vetor de inteiros "parametros"
140                 sscanf(entradas[i + 1], "%d", &parametros[i]);
141             }
142 
143             for (int i = 0; i < 30; ++i)
144             {
145                 if(parametros[i] == -1){
146                     break;
147                 }
148 
149                 else{
150                     qtd_parametros++;
151                 }
152             }
153 
154             instrucao.qtd_parametros = qtd_parametros;
155 
156             // limpa o vetor de instrução da struct
157             for (int i = 0; i < 30; ++i)
158             {
159                instrucao.parametros[i] = -1;
160             }
161 
162             //copiar os parâmetros extraídos para o vetor de parâmetros
163             for (int i = 0; i < 10; ++i)
164             {
165                instrucao.parametros[i] = parametros[i];
166             }
167 
168             interpretar(&instrucao, &ultima_instrucao, &img);
169 
170 
171             //preenche a struct com o comando atual após executá-lo, servindo como um histórico
172 
173             if(strcmp(nome_comando, "line") == 0 || strcmp(nome_comando, "polygon") == 0){
174 
175                 strcpy(ultima_instrucao.nome_comando, nome_comando);
176 
177                 ultima_instrucao.qtd_parametros = qtd_parametros;
178 
179                 // limpa o vetor de instrução da struct
180                 for (int i = 0; i < 30; ++i)
181                 {
182                    ultima_instrucao.parametros[i] = -1;
183                 }
184 
185                 //copiar os parâmetros extraídos para o vetor de parâmetros
186                 for (int i = 0; i < 10; ++i)
187                 {
188                    ultima_instrucao.parametros[i] = parametros[i];
```

```
189                 }
190
191             }
192
193             //limpa os vetores para guardar novos valores
194             for (int i = 0; i < 30; ++i)
195             {
196                 parametros[i] = -1;
197             }
198
199             for (int i = 0; i < 49; ++i)
200             {
201                 text[i] = ' ';
202                 strcpy(entradas[i], " ");
203             }
204             text[49] = ' ';
205             strcpy(entradas[49], " ");
206
207         }
208
209         fclose(file);
210
211         desalocar_matriz(&img);
212     }
213 }
```

5.3.1.2 interpretar()

```
void interpretar (
              comando * entrada,
              comando * ultima_entrada,
              imagem * img )
```

Função de interpretação dos comandos.

Essa função recebe as informações de uma linha de comando representando as primitivas e chama a função da primitiva correspondente com os parâmetros necessários

Parameters

entrada	recebe um ponteiro de uma struct comando com as informações da instrução a ser executada.
ultima_entrada	recebe um ponteiro de uma struct comando com a ultima primitiva salva $*$.
img	recebe um ponteiro dde uma struct imagem com a imagem a ser modificada pelas primitivas

($*$) ultima_entrada somente armazena uma primitiva quando corresponde a line ou polygon, para uso das funções repeat_line e repeat_polygon

Definition at line 24 of file programa.h.

Index