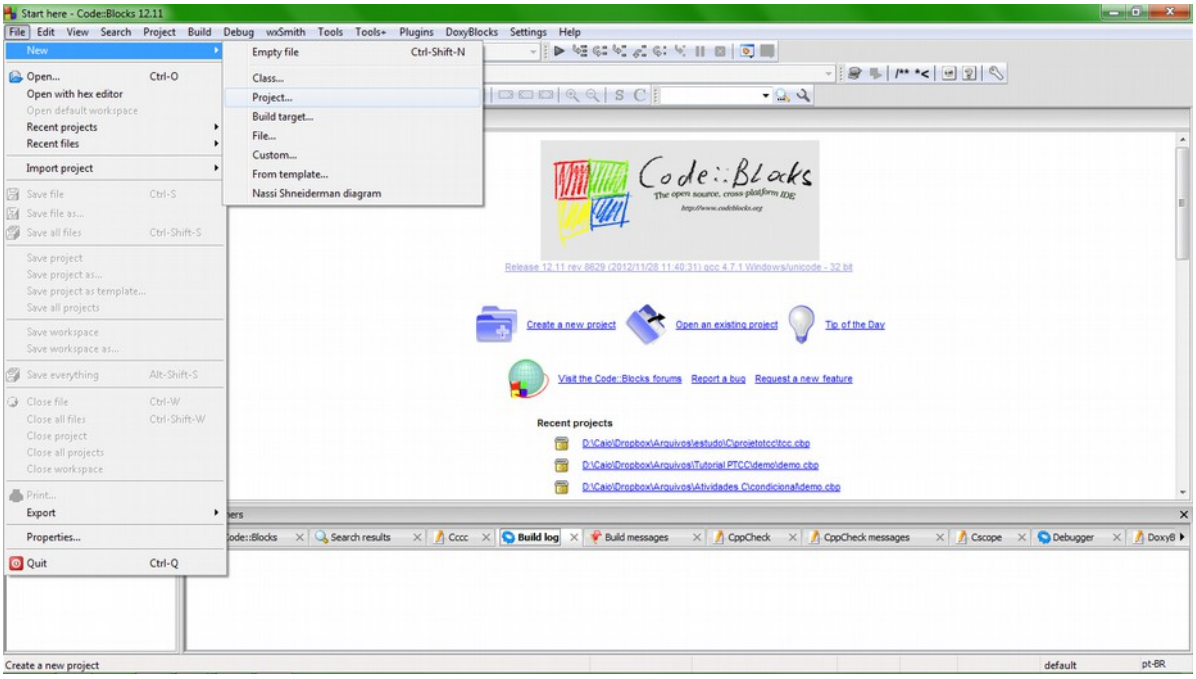
Sumário

1. Começando:

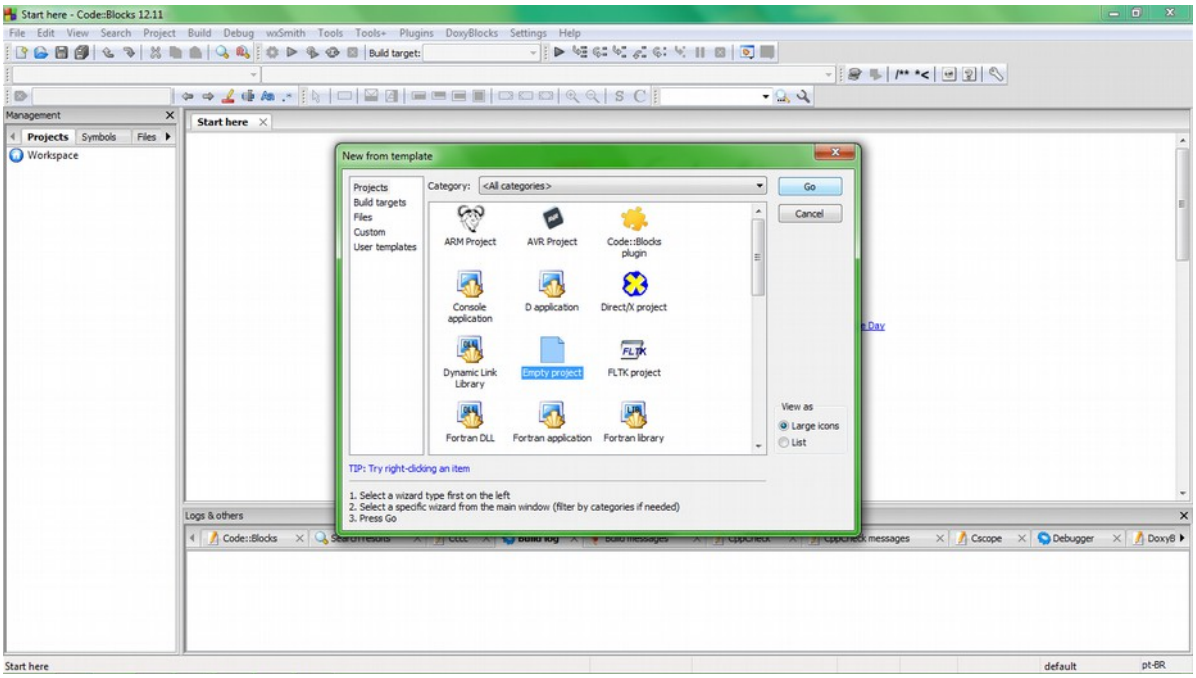
1.1. **Criando seu projeto no Code::Blocks**:

Nesse tutorial ensinaremos um pouco sobre a biblioteca que será usada para a exibição de formas e texto na tela, abaixo será mostrado um passo a passo para a utilização da biblioteca no Code::Blocks:

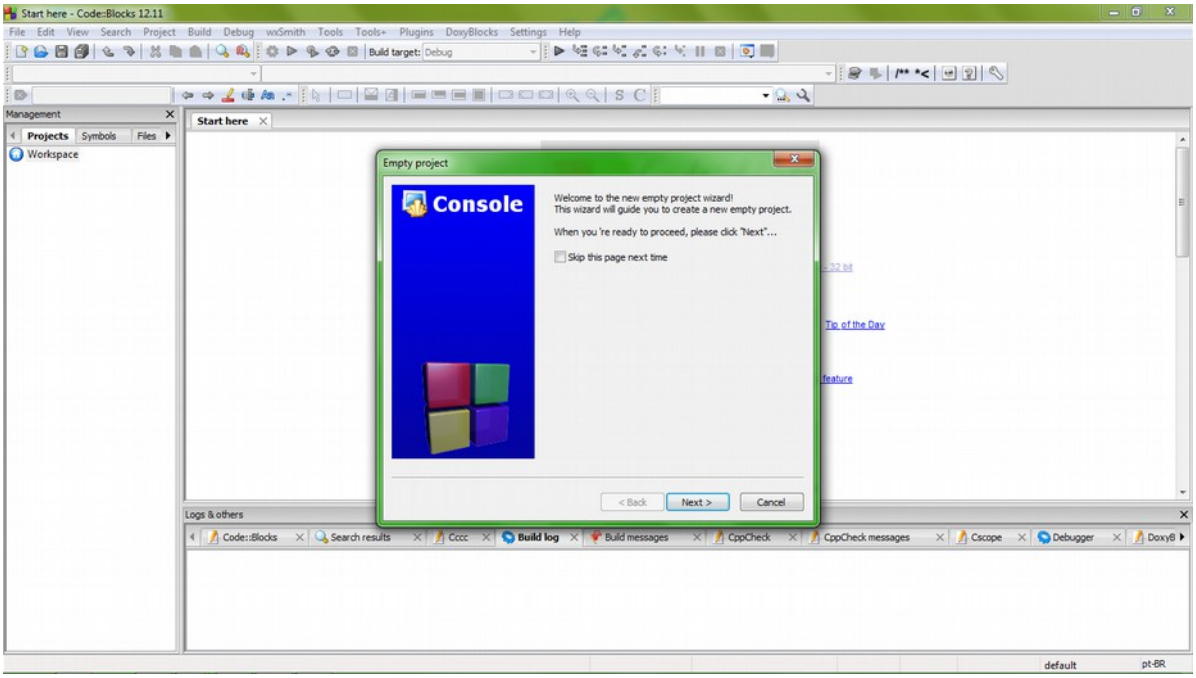
O primeiro passo é selecionar “File” na barra de menus, no menu que abrir selecione “New” e depois “Project”, assim como mostrado na imagem:



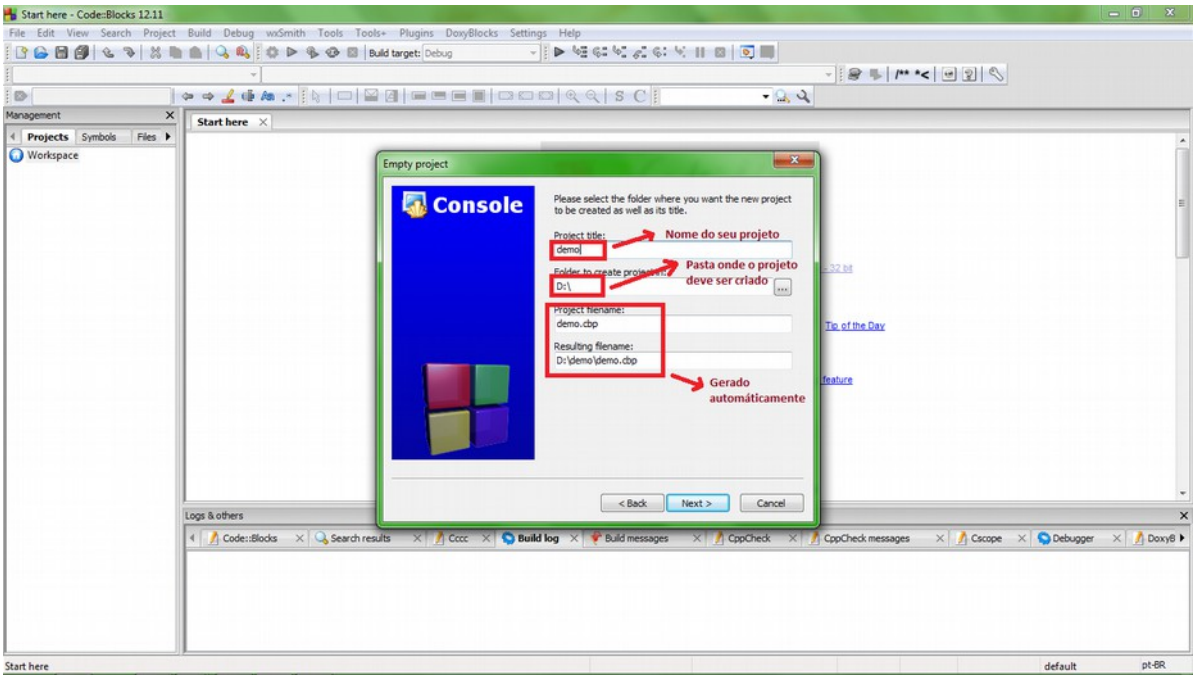
Feito isso, na janela que abrir selecione “Empty project” e clique “Go”.



Uma nova janela abrirá, nessa janela apenas clique em “Next”.

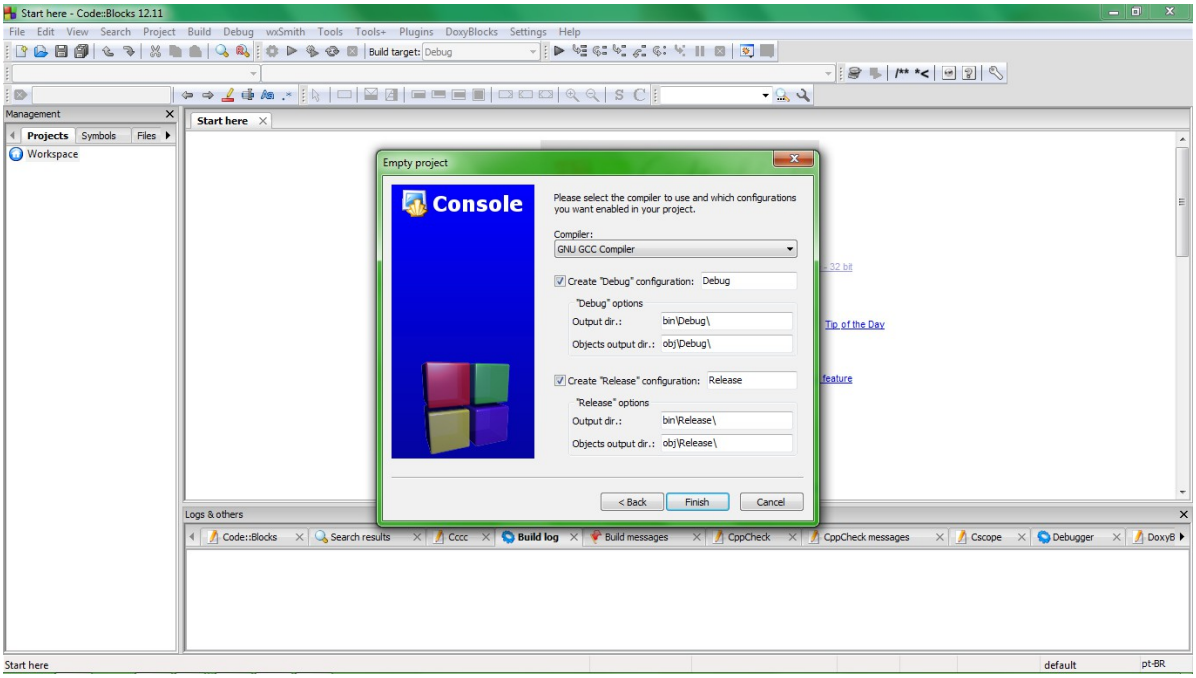


Agora dê um nome ao seu projeto e a pasta onde seu projeto será criado, assim como indicado na imagem abaixo:

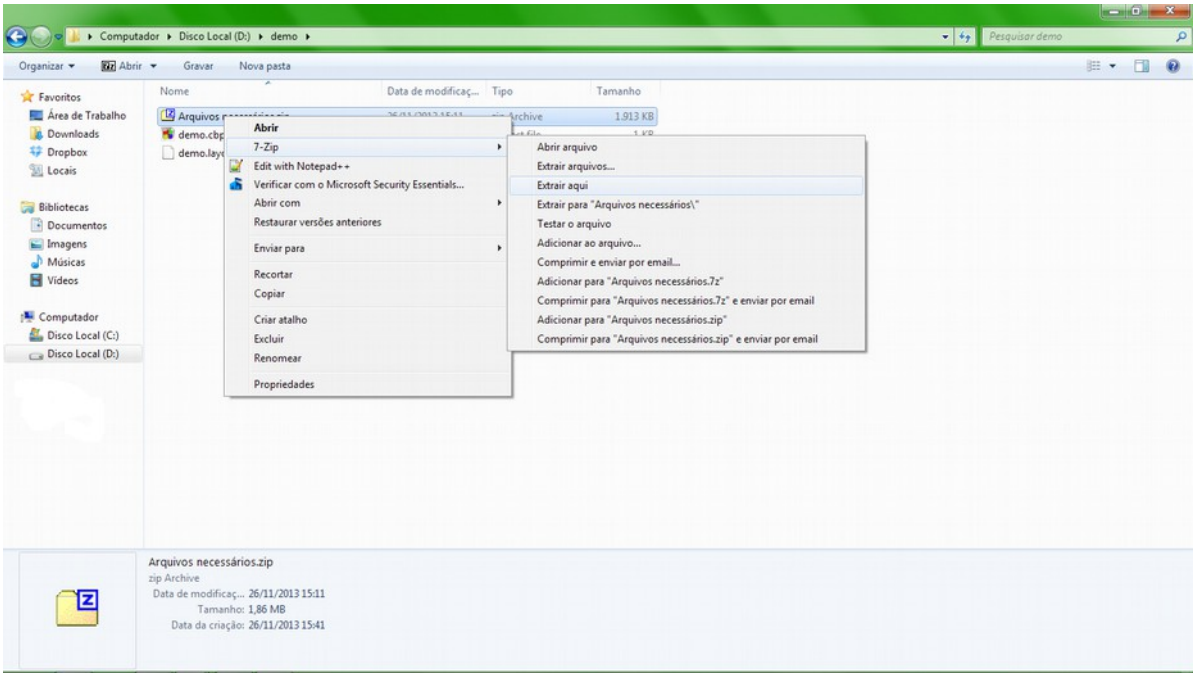


Depois de criar seu projeto, dentro da pasta indicada haverá uma pasta com o mesmo nome de seu projeto e dentro desta pasta terá um arquivo com extensão “.cbp”, esse arquivo é o projeto do Code::Blocks, para abrir seu projeto novamente basta abrir este arquivo.

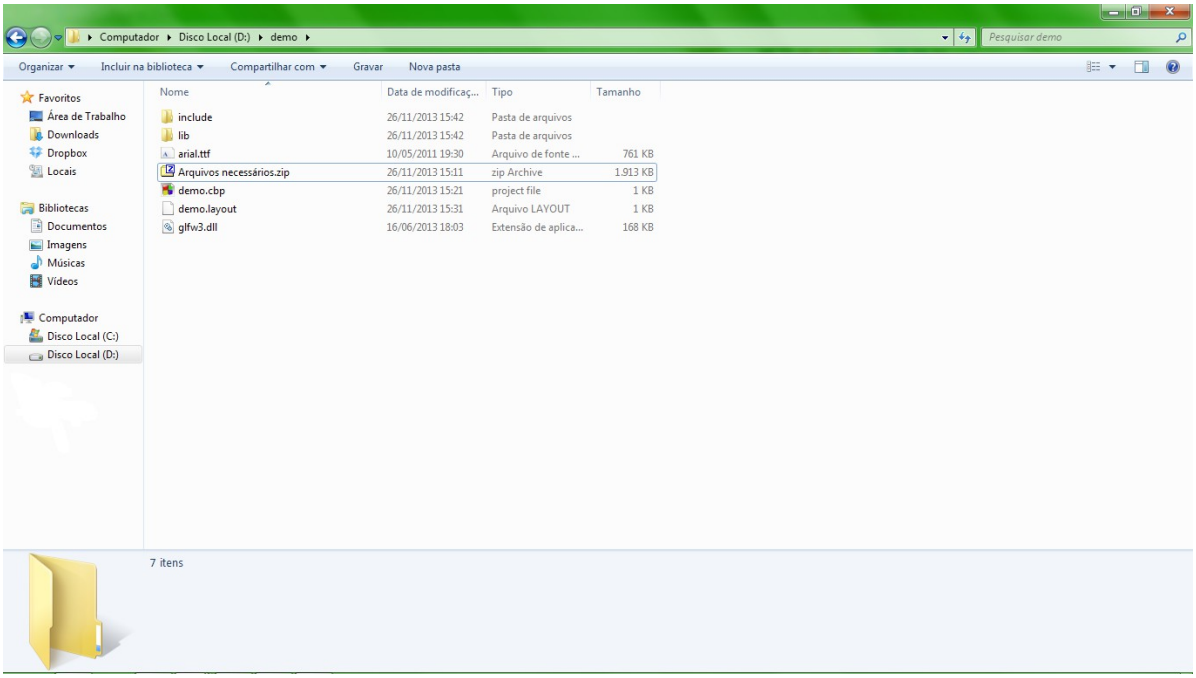
Na próxima tela basta clicar em “Finish”.



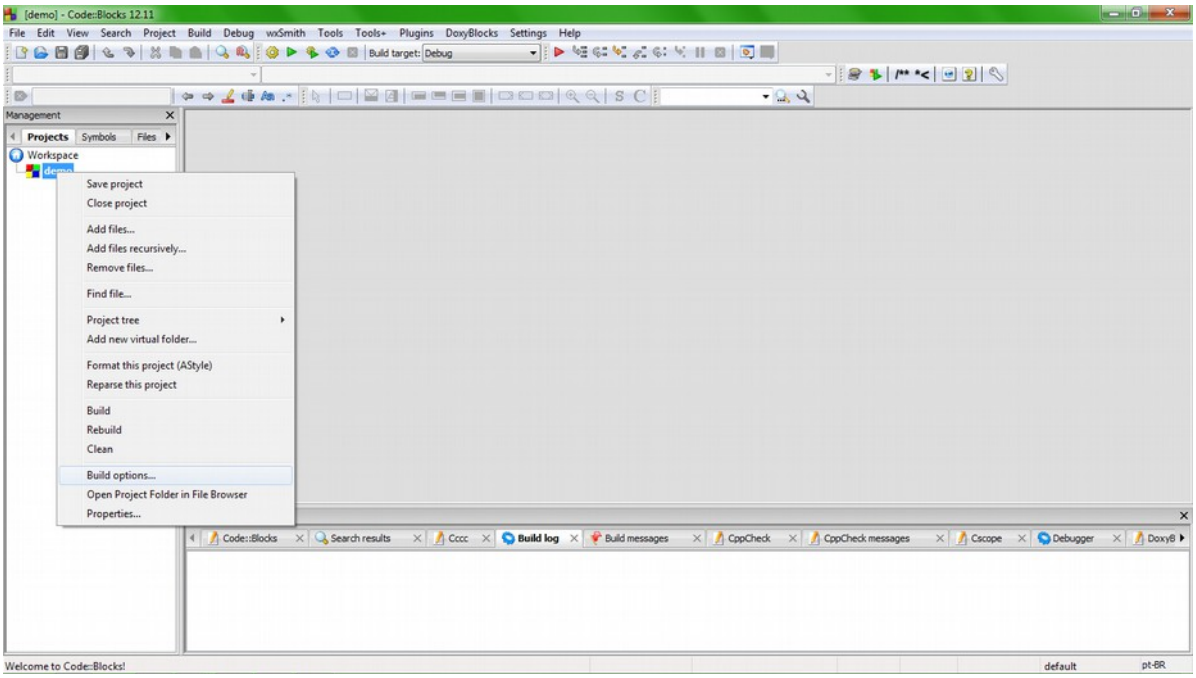
Para usar a biblioteca você deverá passar alguns arquivos para a pasta do projeto, como indicado a seguir:



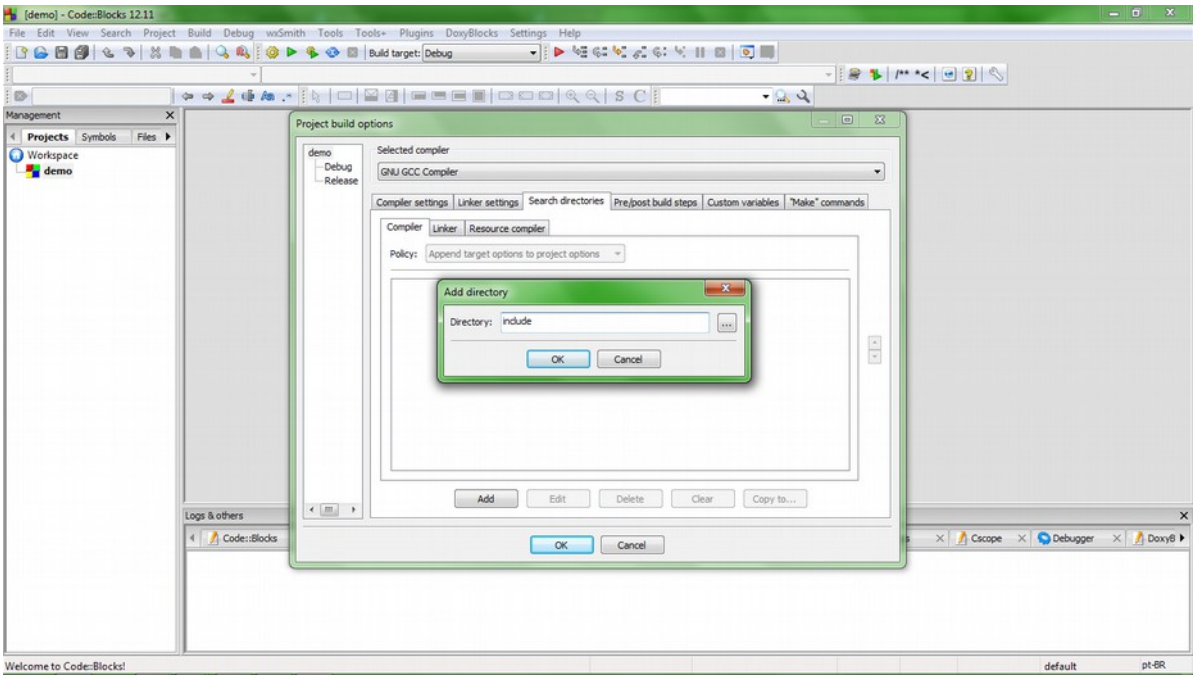
Sua pasta do projeto ficará da seguinte forma:



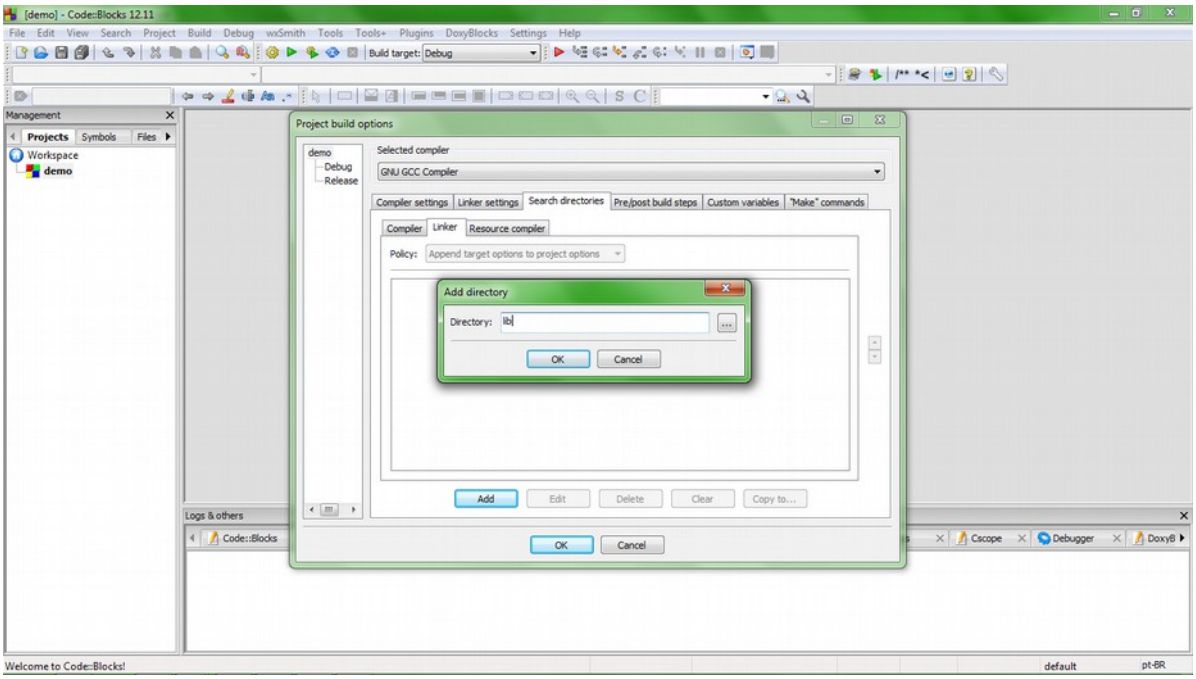
Agora você deve indicar para o Code::Blocks os arquivos que você deseja importar para seu projeto. Primeiro selecione seu projeto e clique com o botão direito do mouse para as opções de projeto apareçam, e selecione “Build options”.



Na lista da esquerda selecione a opção com o nome do seu projeto, selecione “Search directories” e na aba “Compiler” clique em “Add”, na janela que abrir escreva “include” e clique “Ok”.

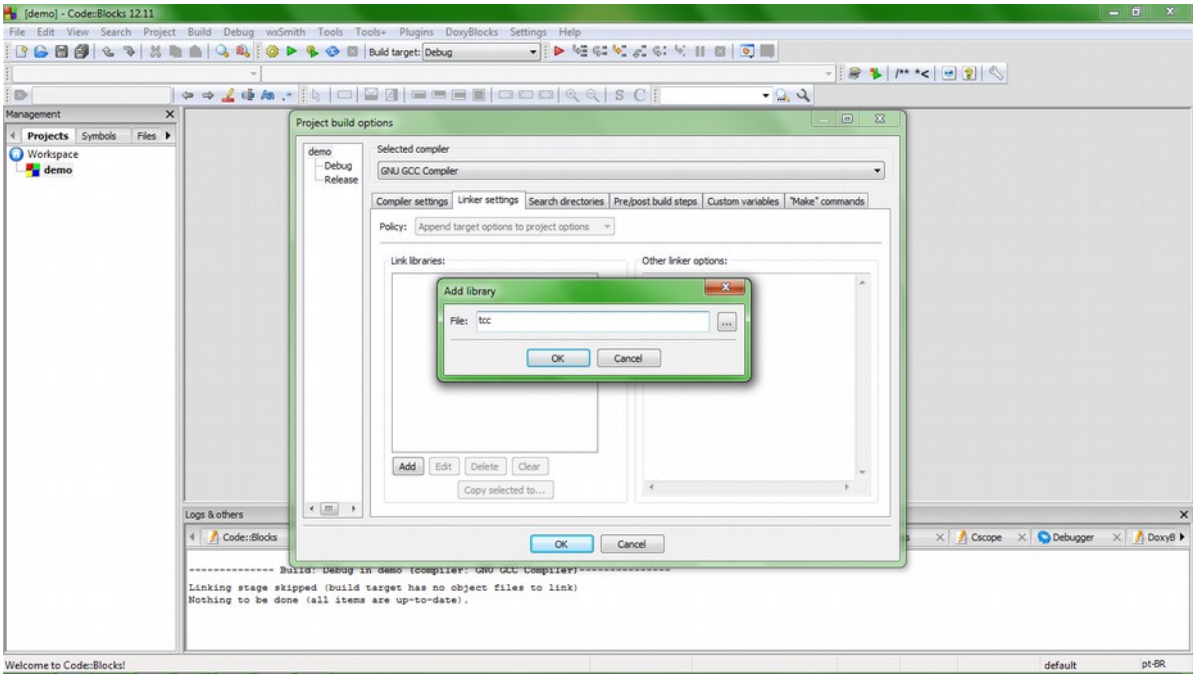


Agora selecione a aba “Linker”, clique em “Add”, na janela que abrir escreva “lib” e clique “Ok”.

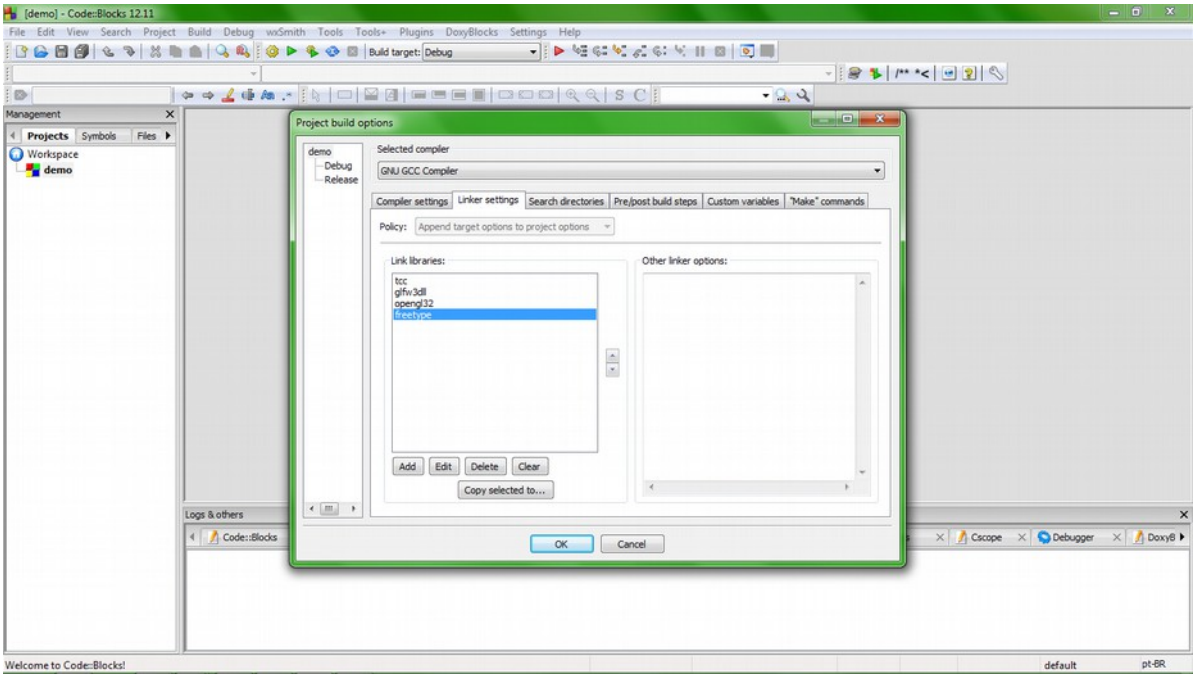


Selecione a aba “Linker settings”, nela clique em “Add”, escreva o nome da biblioteca que deseja adicionar e clique em “Ok”.
Você deverá adicionar as seguintes bibliotecas para seu projeto:

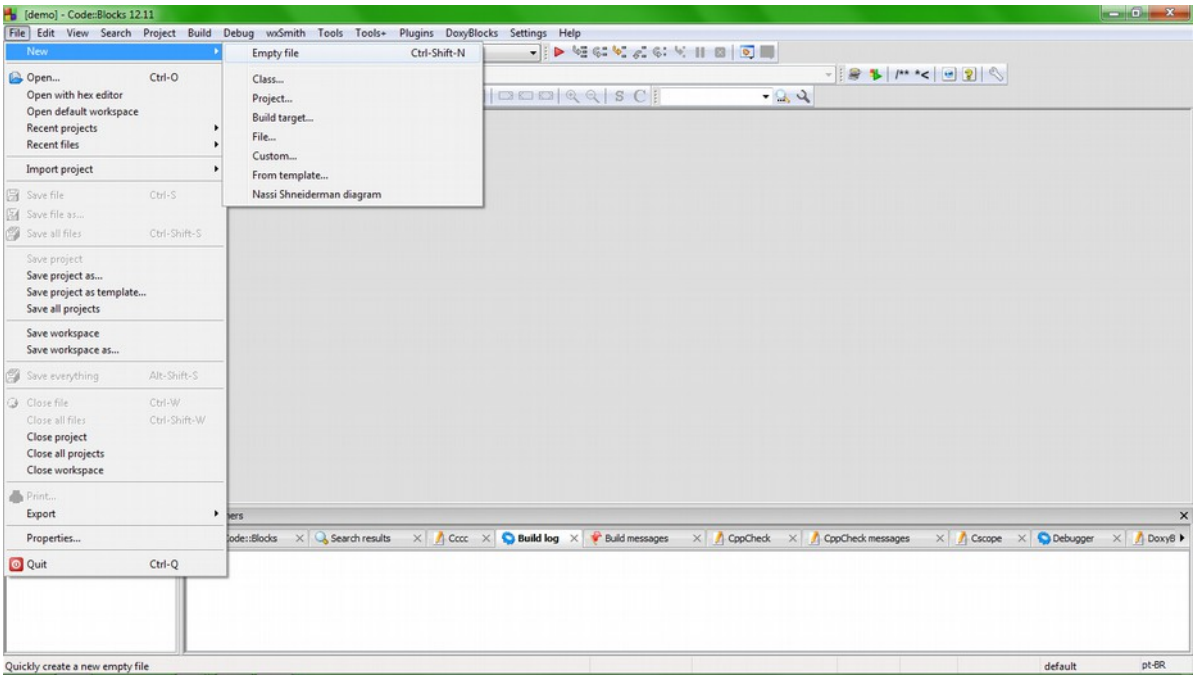
tcc
glfw3dll
opengl32
freetype



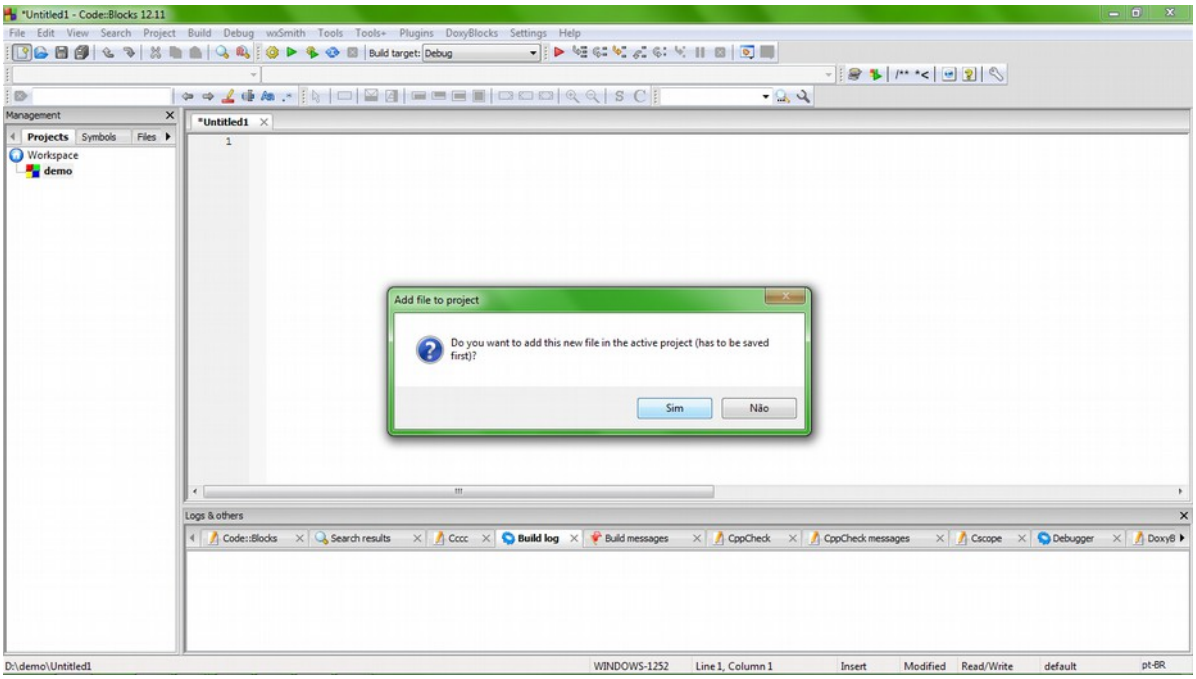
O resultado deve ser este:



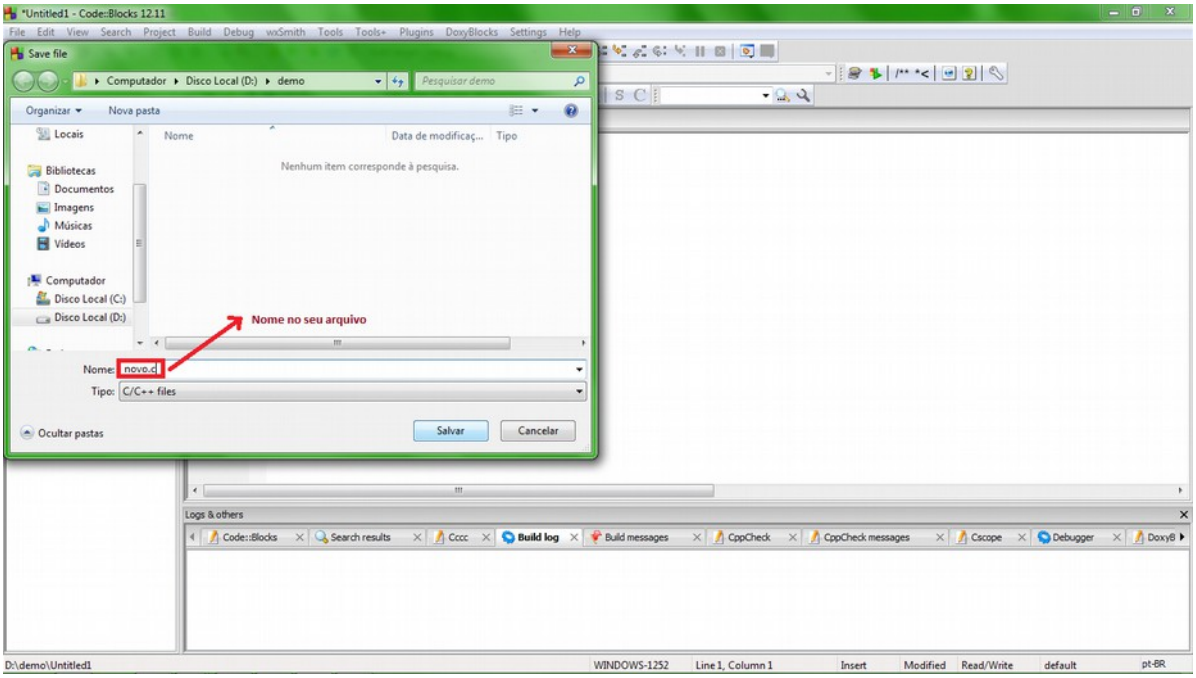
Para começar a programar selecione “File” na barra de menus, no menu que abrir selecione “New” e depois “Empty file”.



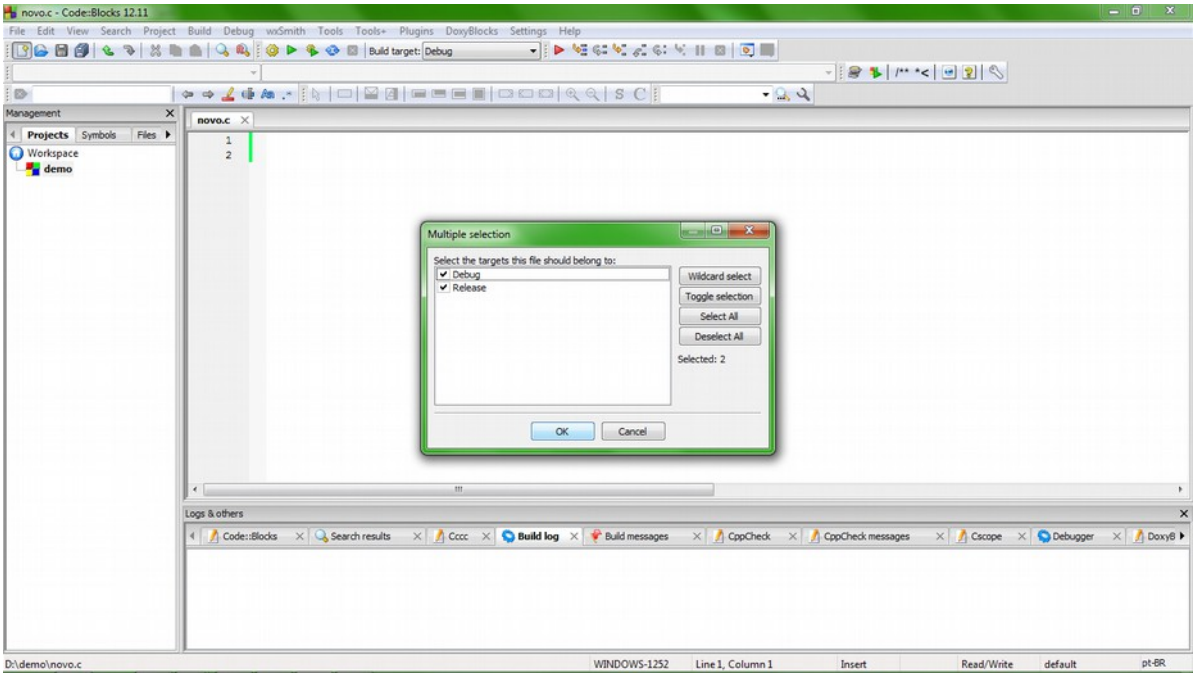
Aparecerá uma mensagem de confirmação, selecione “Sim”.



Agora escolha o nome do arquivo que deseja criar clique em salvar.



Agora clique em “Ok” e comece a programar.



1.2. **Importando a biblioteca**:

Para começar a usar a biblioteca você deve importar o arquivo cabeçalho da biblioteca de funções. Para isso você deve inserir o seguinte código no começo de seu código fonte:

#include <janela.h>

1.3. **Janela**:

Para usar as funcionalidades dessa biblioteca de função será necessário abrir uma janela de desenho. Essa janela será onde as formas posteriormente criadas serão exibidas. Para criar uma janela siga o seguinte exemplo:

janela *jan = criaJanela(800, 600, “Exemplo”);

Nesse exemplo é criada uma janela chamada “jan” de 800 pixels horizontais e 600 pixels verticais, e o título da janela será “Exemplo”.

1.4. **Formas**:

Continuando o exemplo acima, para adicionar formas à sua janela siga o seguinte exemplo:

forma *triang = geraTriangulo(20, 30);
adicionaForma(jan, triang);

Nesse exemplo é gerado um tiângulo e depois o triângulo é inserido na janela.
Na primeira linha é criado uma forma chamada “triang” usando a função “geraTriangulo”, que cria um triângulo de base 20 e altura 30. Mais a frente nesse tutorial serão mostradas as outras funções que geram formas.
Na segunda linha a forma “triang” é inserida na janela “jan” pela função “adicionaForma”, a partir desse momento a forma será exibida na janela e qualquer alteração feita na forma será vista pelo usuário.

2. Lista de Funções:

2.1. **Controle de Janela**:

2.1.1. **janela *criaJanela(int x, int y, char *titulo)**

Descrição: Cria uma janela de desenho;

Entradas:
x – Tamanho da janela na horizontal em pixels;
y – Tamanho da janela na vetical em pixels;
titulo – Nome que será exibido na barra de título;

Saída: Endereço para uma janela com as características especificadas pelas variáveis de entrada. Caso a janela não seja criada com sucesso essa função retornará NULL

2.1.2. **void fechaJanela(janela *j)**

Descrição: Força a janela “j” a fechar, se a janela já foi fechada pelo usuário essa função limpa os dados que foram adicionados a janela;

Entradas:
j – Endereço da janela que deverá ser fechada;

Saída: Nenhuma.

2.1.3. **int adicionaForma(janela *j, forma *f)**

Descrição: Adiciona a forma “f” à janela “j”;

Entradas:
j – Endereço da janela onde a forma será inserida;
f – Endereço da forma que será inserida na janela;

Saída: Valor referente ao resultado, se 0 a forma foi adicionada com sucesso, se -1 a forma não pôde ser inserida.

2.1.4. **int removeForma(janela *j, forma *f)**

Descrição: Remove a forma “f” da janela “j”;

Entradas:
j – Endereço da janela onde a forma deve ser removida;
f – Endereço da forma que deve ser removida da janela;

Saída: Valor referente ao resultado, se 0 a forma foi removida com sucesso, se -1 a forma não pôde ser removida.

2.1.5. **int limpaJanela(janela *j)**

Descrição: Remove todas as formas inseridas na janela “j”;

Entradas:
j – Endereço da janela que será limpa;

Saída: Valor referente ao resultado, se 0 as formas foram removidas com sucesso, se -1 nenhuma forma pôde ser removida.

2.1.6. **void mudaCorFundo(janela *j, GLubyte vermelho, GLubyte verde, GLubyte azul)**

Descrição: Muda a cor que a janela “j” possui de fundo;

Entradas:
j – Endereço da janela que terá sua cor de fundo alterada;
vermelho – Valor de 0 à 255 que define a intensidade do vermelho na cor;
verde – Valor de 0 à 255 que define a intensidade do verde na cor;
azul – Valor de 0 à 255 que define a intensidade do azul na cor;

Saída: Nenhuma.

2.1.7. **void mudaEspecial(janela *j, void (*especial)())**

Descrição: Muda a função que será chamada durante a atualização da tela de desenho;

Entradas:
j – Endereço da janela que terá sua função especial alterada;
especial – Endereço para a função especial;

Saída: Nenhuma.

2.2. **Controle de Forma**:

2.2.1. **void definePosicao(double x, double y)**

Descrição: Muda a posição que as próximas formas serão geradas, a posição padrão é o centro;

Entradas:
x – Posição horizontal em pixels contada como -t / 2 na extremidade da esquerda, 0 no centro e t / 2 na extremidade da direita, sendo t o tamanho horizontal da tela;
y – Posição vertical em pixels contada como -t / 2 na extremidade de baixo, 0 no centro e t / 2 na extremidade de cima, sendo t o tamanho vertical da tela:

Saída: Nenhuma.

2.2.2. **void defineTamanhoBorda(double t)**

Descrição: Muda o tamanho do ponto, linha e contorno que as próximas formas serão geradas, o tamanho padrão é 1 pixel;

Entradas:
t – Tamanho em pixel do ponto, linha ou contorno;

Saída: Nenhuma.

2.2.3. **void defineCorInterna(GLubyte vermelho, GLubyte verde, GLubyte azul)**

Descrição: Muda a cor de preenchimento das próximas formas sólidas que forem geradas, a cor padrão é o branco;

Entradas:
vermelho – Valor de 0 à 255 que define a intensidade do vermelho na cor;
verde – Valor de 0 à 255 que define a intensidade do verde na cor;
azul – Valor de 0 à 255 que define a intensidade do azul na cor;

Saída: Nenhuma.

2.2.4. **void defineCorBorda(GLubyte vermelho, GLubyte verde, GLubyte azul)**

Descrição: Muda a cor que os próximos pontos e linhas serão geradas, também a cor do contorno das próximas formas que serão geradas, a cor padrão é branco;

Entradas:

vermelho – Valor de 0 à 255 que define a intensidade do vermelho na cor;
verde – Valor de 0 à 255 que define a intensidade do verde na cor;
azul – Valor de 0 à 255 que define a intensidade do azul na cor;

Saída: Nenhuma.

2.2.5. **forma *geraPonto()**

Descrição: Cria um ponto;

Entradas:

Nenhuma;

Saída: Endereço da forma gerada.

Exemplo:

2.2.6. **forma *geraLinha(double tamanho)**

Descrição: Cria uma linha horizontal;

Entradas:

tamanho – Tamanho da linha em pixels;

Saída: Endereço da forma gerada.

Exemplo:

2.2.7. **forma *geraTriangulo(double base, double altura)**

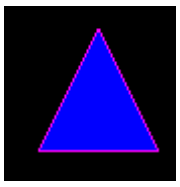
Descrição: Cria um triângulo sólido;

Entradas:

base – Tamanho da base do triângulo em pixels;
altura – Altura do triângulo em pixels;

Saída: Endereço da forma gerada.

Exemplo:



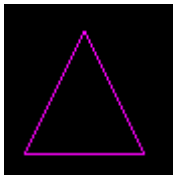
2.2.8. **forma *geraTrianguloVazado(double base, double altura)**

Descrição: Cria um triângulo vazado;

Entradas:
base – Tamanho da base do triângulo em pixels;
altura – Altura do triângulo em pixels;

Saída: Endereço da forma gerada.

Exemplo:



2.2.9. **forma *geraRetangulo(double base, double altura)**

Descrição: Cria um retângulo sólido;

Entradas:
base – Tamanho da base do retângulo em pixels;
altura – Altura do retângulo em pixels;

Saída: Endereço da forma gerada.

Exemplo:

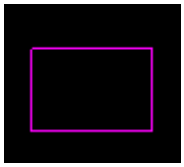
2.2.10. **forma *geraRetanguloVazado(double base, double altura)**

Descrição: Cria um retângulo vazado;

Entradas:
base – Tamanho da base do retângulo em pixels;
altura – Altura do retângulo em pixels;

Saída: Endereço da forma gerada.

Exemplo:



2.2.11. **forma *geraCirculo(double raio)**

Descrição: Cria um círculo sólido;

Entradas:
raio – Raio do circulo em pixels;

Saída: Endereço da forma gerada.

Exemplo:

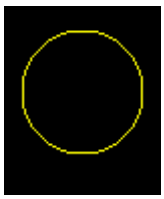
2.2.12. **forma *geraCirculoVazado(double raio)**

Descrição: Cria um círculo vazado;

Entradas:
raio – Raio do circulo em pixels;

Saída: Endereço da forma gerada.

Exemplo:



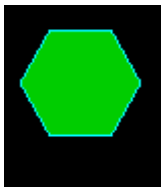
2.2.13. **forma *geraPoligono(double raio, int nVertices)**

Descrição: Cria um polígono regular sólido;
Ex.:
nVertices = 3 => triângulo equilátero;
nVertices = 4 => quadrado;
nVertices = 5 => pentágono;
nVertices = 6 => exágono;

Entradas:
raio – Distância do centro do polígono até os vértices em pixels;
nVertices – Número de vértices presentes no polígono;

Saída: Endereço da forma gerada.

Exemplo:



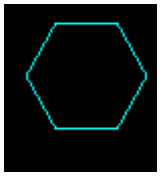
2.2.14. **forma *geraPoligonoVazado(double raio, int nVertices)**

Descrição: Cria um polígono regular vazado;
Ex.:
nVertices = 3 => triângulo equilátero;
nVertices = 4 => quadrado;
nVertices = 5 => pentágono;
nVertices = 6 => exágono;

Entradas:
raio – Distância do centro do polígono até os vértices em pixels;
nVertices – Número de vértices presentes no polígono;

Saída: Endereço da forma gerada.

Exemplo:



2.2.15. **forma *geraElipse(double raio1, double raio2)**

Descrição: Cria uma elipse sólida;

Entradas:
raio1 – Raio horizontal da elipse em pixels;
raio2 – Raio vertical da elipse em pixels;

Saída: Endereço da forma gerada.

Exemplo:

2.2.16. **forma *geraElipseVazado(double raio1, double raio2)**

Descrição: Cria uma elipse vazada;

Entradas:
raio1 – Raio horizontal da elipse em pixels;
raio2 – Raio vertical da elipse em pixels;

Saída: Endereço da forma gerada.

Exemplo:

2.2.17. **forma *geraSetor(double raio, int angulo)**

Descrição: Cria um setor sólido;

Entradas:
raio – Raio do setor em pixels;
angulo – ângulo do setor em graus;

Saída: Endereço da forma gerada.

Exemplo:

2.2.18. **forma *geraSetorVazado(double raio, int ângulo)**

Descrição: Cria um setor vazado;

Entradas:
raio – Raio do setor em pixels;
angulo – ângulo do setor em graus;

Saída: Endereço da forma gerada.

Exemplo:

2.2.19. **forma *geraArco(double raio, int ângulo)**

Descrição: Cria um arco sólido;

Entradas:
raio – Raio do arco em pixels;
angulo – ângulo do arco em graus;

Saída: Endereço da forma gerada.

Exemplo:

2.2.20. **forma *geraArcoVazado(double raio, int ângulo)**

Descrição: Cria um arco vazado;

Entradas:
raio – Raio do arco em pixels;
angulo – ângulo do arco em graus;

Saída: Endereço da forma gerada.

Exemplo:

2.2.21. **forma *geraArcoAberto(double raio, int ângulo)**

Descrição: Cria um arco aberto;

Entradas:
raio – Raio do arco em pixels;
angulo – ângulo do arco em graus;

Saída: Endereço da forma gerada.

Exemplo:

2.2.22. **forma *geraEstrela(double raio1, double raio2, int nPontas)**

Descrição: Cria uma estrela sólida;

Entradas:
raio1 – Raio dos vértices ímpares em pixels;
raio2 – Raio dos vértices pares em pixels;
nPontas – número de pontas que a estrela terá;

Saída: Endereço da forma gerada.

Exemplo:

2.2.23. **forma *geraEstrelaVazado(double raio1, double raio2, int nPontas)**

Descrição: Cria uma estrela vazada;

Entradas:
raio1 – Raio dos vértices ímpares em pixels;
raio2 – Raio dos vértices pares em pixels;
nPontas – número de pontas que a estrela terá;

Saída: Endereço da forma gerada.

Exemplo:

2.2.24. **forma *geraPentagrama(double raio1, double raio2)**

Descrição: Cria um pentagrama sólido;

Entradas:
raio1 – Raio dos vértices ímpares em pixels;
raio2 – Raio dos vértices pares em pixels;

Saída: Endereço da forma gerada.

Exemplo:

2.2.25. **forma *geraPentagramaVazado(double raio1, double raio2)**

Descrição: Cria um pentagrama vazado;

Entradas:
raio1 – Raio dos vértices ímpares em pixels;
raio2 – Raio dos vértices pares em pixels;

Saída: Endereço da forma gerada.

Exemplo:

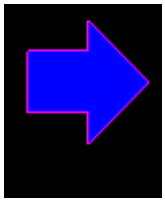
2.2.26. **forma *geraSeta(double proporcao)**

Descrição: Cria uma seta sólida;

Entradas:
proporcao – proporção que a seta será gerada, a setá terá o dobro em pixels em suas extremidades horizontais e verticais;

Saída: Endereço da forma gerada.

Exemplo:



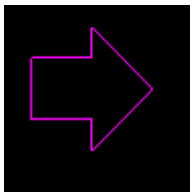
2.2.27. **forma *geraSetaVazado(double proporcao)**

Descrição: Cria uma seta vazada;

Entradas:
proporcao – proporção que a seta será gerada, a seta terá o dobro em pixels em suas extremidades horizontais e verticais;

Saída: Endereço da forma gerada.

Exemplo:



2.2.28. **forma *geraTexto(double tamanho, char *texto)**

Descrição: Cria um texto;

Entradas:
tamanho – proporção que o texto será gerado;
texto – texto que se deseja gerar na tela;

Saída: Endereço da forma gerada.

Exemplo:

2.2.29. **int destroiForma(forma *f)**

Descrição: Limpa a memória alocada para a forma “f”;

Entradas:
f – forma que deverá ser destruída;

Saída: Valor referente ao resultado, se 0 a forma foi destruída com sucesso, se -1 a forma não pôde ser destruída.

2.2.30. **void rodar(forma *f, double a)**

Descrição: Roda a forma “f”;

Entradas:
f – forma que será rotacionada;
a – ângulo que a forma deve ser rotacionada em graus;

Saída: Nenhuma.

2.2.31. **void redimensionar(forma *f, double x, double y)**

Descrição: Redimensiona a forma “f”;

Entradas:
f – forma que será redimensionada;
x – quantas vezes o tamanho no eixo x da forma será multiplicado;
y – quantas vezes o tamanho no eixo y da forma será multiplicado;

Saída: Nenhuma.

2.2.32. **void mover(forma *f, double x, double y)**

Descrição: Move a forma “f”;

Entradas:
f – forma que será movida;
x – quantos pixels a forma será movida no eixo x;
y – quantos pixels a forma será movida no eixo y;

Saída: Nenhuma.

2.2.33. **void moverPara(forma *f, double x, double y)**

Descrição: Move a forma “f” para a posição estipulada;

Entradas:
f – forma que será movida;
x – o ponto em pixels para onde a forma será movida no eixo x;
y – o ponto em pixels para onde a forma será movida no eixo y;

Saída: Nenhuma.

2.2.34. **void mudaTexto(forma *f, double tamanho, char *texto)**

Descrição: Muda o texto da forma “f”, se a forma não for um texto ela será substituída por um texto;

Entradas:
f – forma que será alterada;
tamanho – proporção que o texto será gerado;
texto – texto que se deseja gerar na tela;

Saída: Nenhuma.

2.2.35. **void mudaTamanhoBorda(forma *f, double t)**

Descrição: Muda o tamanho da borda da forma “f”;

Entradas:
f – forma que será alterada;
tamanho – proporção que o texto será gerado;
texto – texto que se deseja gerar na tela;

Saída: Nenhuma.

2.2.36. **void mudaCorInterna(forma *f, GLubyte vermelho, GLubyte verde, GLubyte azul)**

Descrição: Muda a cor interna da forma “f”;

Entradas:
f – forma que será alterada;
vermelho – Valor de 0 à 255 que define a intensidade do vermelho na cor;
verde – Valor de 0 à 255 que define a intensidade do verde na cor;
azul – Valor de 0 à 255 que define a intensidade do azul na cor;

Saída: Nenhuma.

2.2.37. **void mudaCorBorda(forma *f, GLubyte vermelho, GLubyte verde, GLubyte azul)**

Descrição: Muda a cor da borda da forma “f”;

Entradas:
f – forma que será alterada;
vermelho – Valor de 0 à 255 que define a intensidade do vermelho na cor;
verde – Valor de 0 à 255 que define a intensidade do verde na cor;
azul – Valor de 0 à 255 que define a intensidade do azul na cor;

Saída: Nenhuma.