

プログラミング基礎 #09

変数と配列2

担当：向井 智彦

本日の内容

- 目標：変数と配列について理解を深める
- 講義内容：
 - 「変数と配列」おさらい
 - グラフィクス描画における変数・配列の利用
- 演習：線を駆使した2次元CGプログラミング

「変数と配列」まとめ

- 変数：1つのデータを一時保存できる
 - 変数宣言 「int a;」, 代入(上書き) 「a = 10;」
 - 変数からの代入(読み出し) 「int b = a;」
- 配列：同じ型の複数の変数を添字付きで格納
 - 配列宣言 「int a[10];」
 - 配列の先頭番号は0 「a[0] = 1;」...「a[9] = 10;」
- 変数を用いた演算
 - 「a = a + b + 10 * a;」など代入演算子 = の左右に同じ変数が登場する際の動作

変数と演算

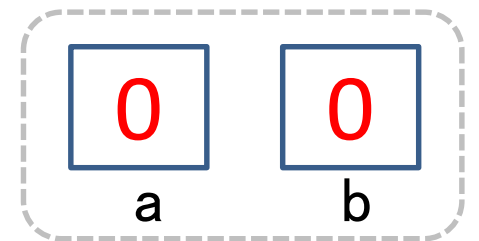
以下のプログラムの出力は？

```
int a = 0, b = 0; // 2つまとめて宣言 & 初期化
a = 10;
b = a;
a = a + 20;
cout << a << ", " << b << endl;
```

変数と演算

1. 変数 a と b を, 初期値 0 で作成

```
int a = 0, b = 0;  
a = 10;  
b = a;  
a = a + 20;  
cout << a << ", " << b << endl;
```



変数と演算

2. 変数 a に値 10 が代入される

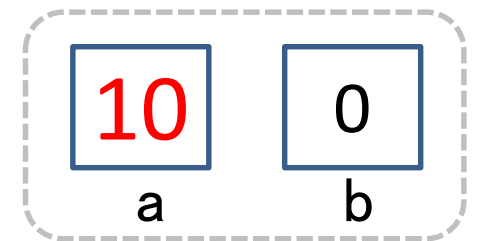
```
int a = 0, b = 0;
```

```
a = 10;
```

```
b = a;
```

```
a = a + 20;
```

```
cout << a << ", " << b << endl;
```



変数と演算

3. 1. 変数 a が保持する値 10 を読み出し

```
int a = 0, b = 0;
```

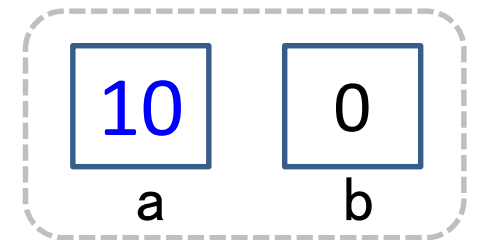
```
a = 10;
```

```
b = a,
```

10

```
a = a + 20;
```

```
cout << a << ", " << b << endl;
```



変数と演算

3. 2. 変数 b に変数 a が保持している値 10 が代入される

```
int a = 0, b = 0;
```

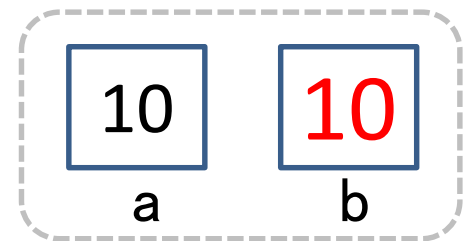
```
a = 10;
```

```
b = a;
```

10

```
a = a + 20;
```

```
cout << a << ", " << b << endl;
```



変数と演算

4. 1. 変数 a が保持している値 10 を読み出し

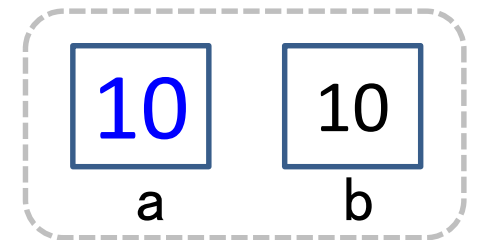
```
int a = 0, b = 0;
```

```
a = 10;
```

```
b = a;
```

```
a = a + 20;
```

```
cout << a << ", " << b << endl;
```



変数と演算

4. 2. 読み出した値 10 と 20 を加算

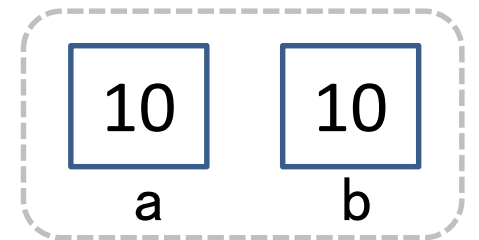
```
int a = 0, b = 0;
```

```
a = 10;
```

```
b = a;
```

```
a = a + 20;
```

```
cout << a << ", " << b << endl;
```



10

変数と演算

4. 3. 加算結果 30 を変数 a に代入

```
int a = 0, b = 0;
```

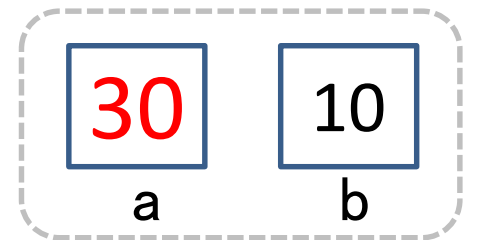
```
a = 10;
```

```
b = a;
```

```
a = a + 20;
```

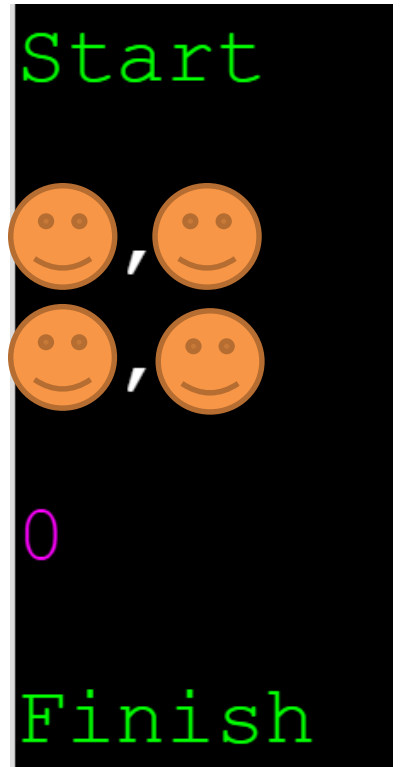
```
cout << a << ", " << b << endl;
```

30



クイズ： 出力結果は？

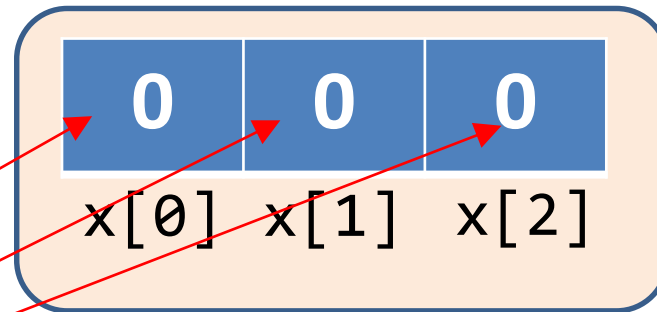
```
#include <iostream>
using namespace std;
int main()
{
    int x = 10, y = 20;
    cout << x << "," << y << endl;
    x = 30;
    x = y + x;
    y = x - 10;
    cout << x << "," << y << endl;
}
```



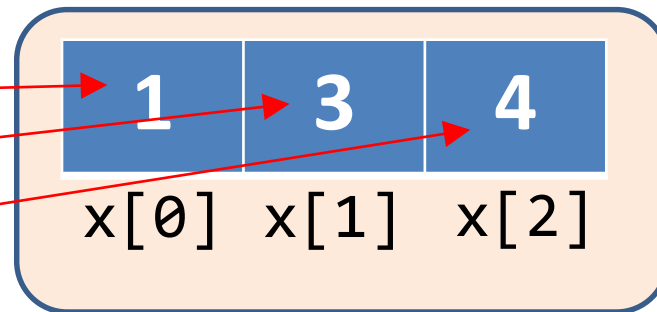
配列

- 複数の変数を番号付けて並べる(最初は0番)

```
#include <iostream>
using namespace std;
int main()
{
```



↓ 代入



```
    int x[3] = {0, 0, 0};
    x[0] = 1;
    x[1] = 3;
    x[2] = 4;
    cout << x[0] << " " << x[1] << "," << x[2] << endl;
}
```

使い分けワンポイント

- 同種類・同意味の複数の値をまとめて扱う時には配列を利用
- その他の場合は変数を利用
 - 1つの値だけを用いる場合
 - 異なる種類・異なる意味を持つ複数の値

ワンポイントの例1

望ましくない書き方

```
double karada[4];  
karada[0] = 1.7; //身長  
karada[1] = 65.0; //体重  
karada[2] = 18.5; //体脂肪  
karada[3] = 75.0; //胴回り
```

異なる種類・異なる意味の数字
が1つの配列にまとめられている

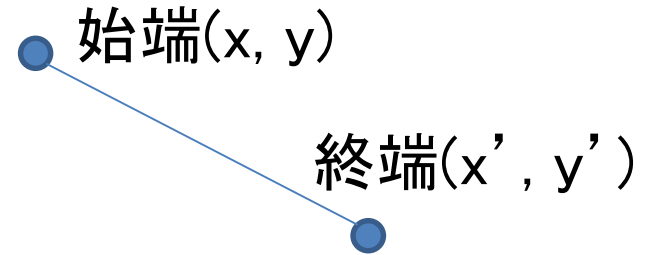
望ましい書き方

```
double shincho;  
double weight;  
double bodyfat;  
double waist;
```

異なる意味・単一の数値が
各1つの変数で表されている
(double shicho[1];なども冗長なので避ける)

※ちなみに本例はクラスを用いるのが適当

ワンポイントの例2



望ましくない書き方

```
double pos[4];  
pos[0] = -40.5; //始端x  
pos[1] = 65.0; //終端y  
pos[2] = 12.5; //始端x'  
pos[3] = -5.0; //終端y'
```

異なる種類・異なる意味の数字
が1つの配列にまとめられている

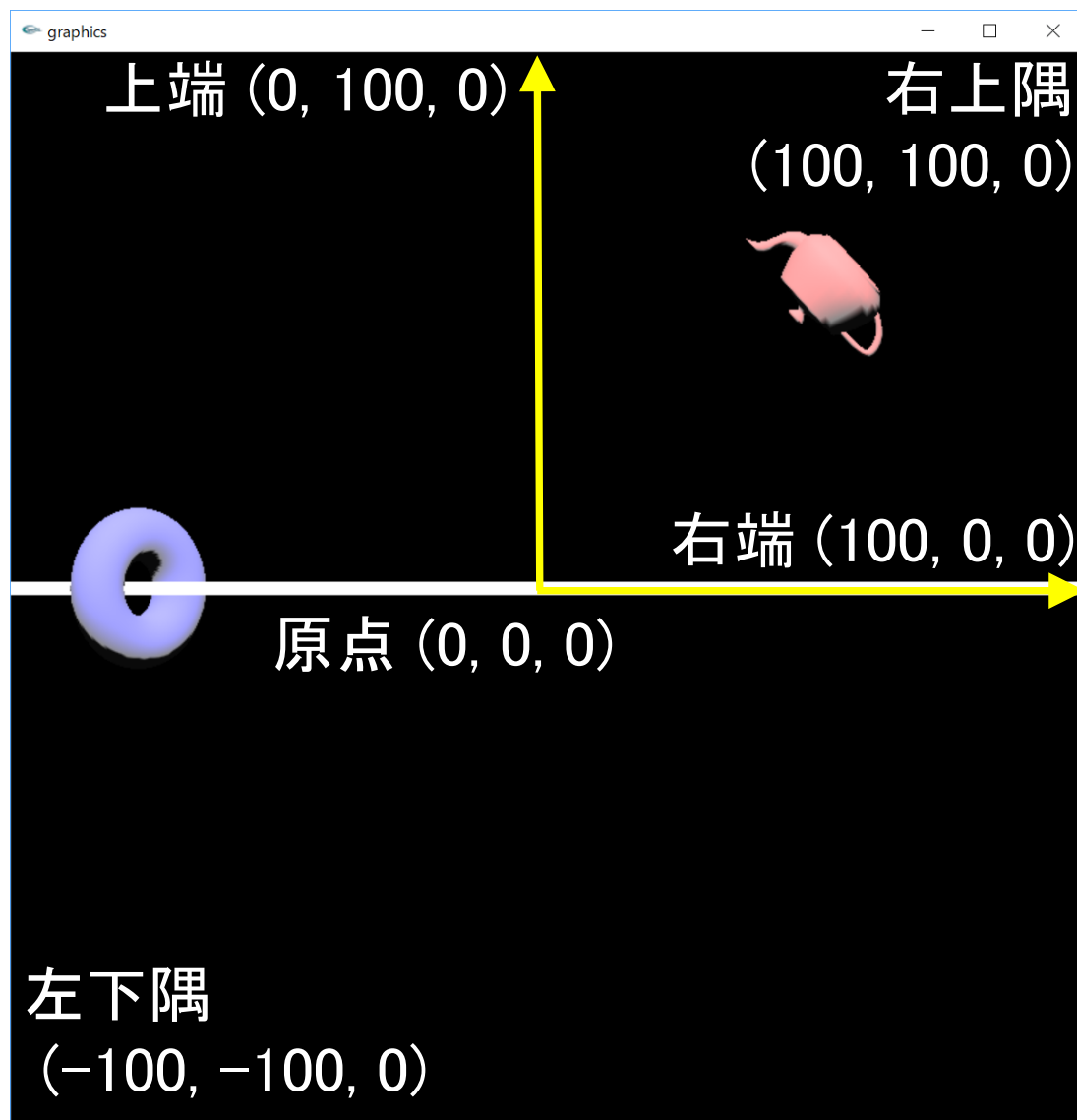
望ましい書き方

```
double shitan[2];  
double shutan[2];  
あるいは  
double xpos[2];  
double ypos[2];
```

同じ種類・同じ意味の数値を
それぞれ配列にまとめる
(shitan_x, ... shutan_y のように
4つの変数にバラしても可)

※ちなみに本例もクラスを用いるのが適当

グラフィックスアプリの座標系



線の描き方 (display関数内”2Dパート”に記述)

// 線の太さ

```
glLineWidth(10.0); // これ以降に描く線は全て太さ 10
```

// 線の色(R, G, B)

```
glColor3d(1.0, 1.0, 1.0); // これ以降の頂点は全て白
```

// 線の描画

```
glBegin(GL_LINES); // 線描はじめ
```

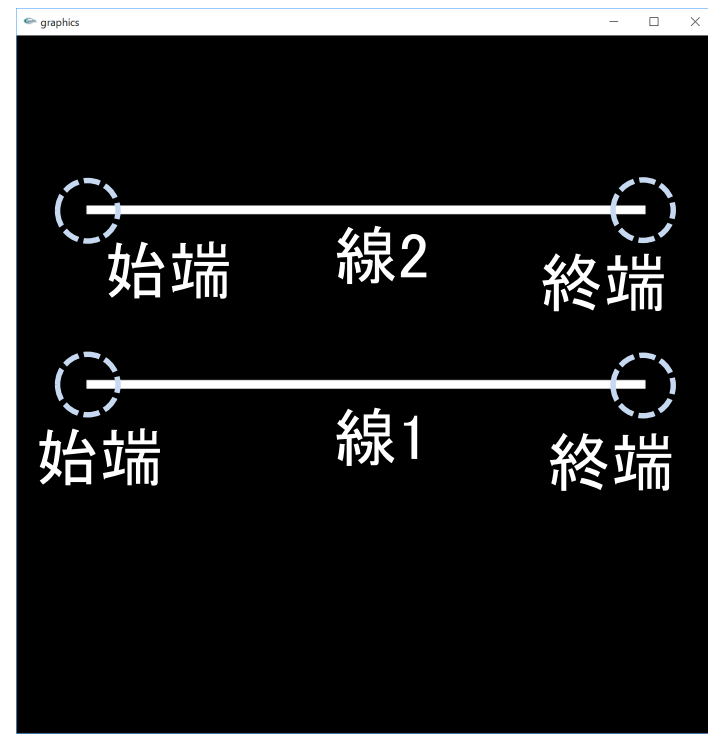
```
glVertex3d(-80.0, 0, 0); // 線1の始端
```

```
glVertex3d( 80.0, 0, 0); // 線1の終端
```

```
glVertex3d(-80.0, 50, 0); // 線2の始端
```

```
glVertex3d( 80.0, 50, 0); // 線2の終端
```

```
glEnd(); // 線描おわり
```

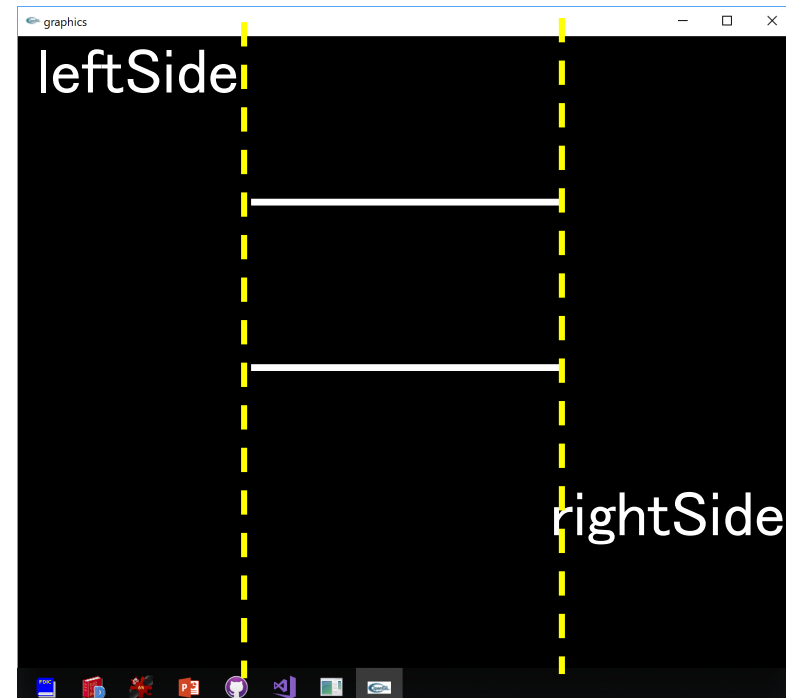


デモ

- 3本の線を用いて四角形を描く
- 4本の線を用いて四角形を描く
- 多くの平行な線を描く
- 1本の線の両端で異なる色を指定する

例：同じ位置を表す変数

```
double leftSide = -40.0; // 左端  
double rightSide = 40.0; // 右端  
glBegin(GL_LINES);  
glVertex3d(leftSide, 0, 0);  
glVertex3d(rightSide, 0, 0);  
  
glVertex3d(leftSide, 50, 0);  
glVertex3d(rightSide, 50, 0);  
glEnd();
```

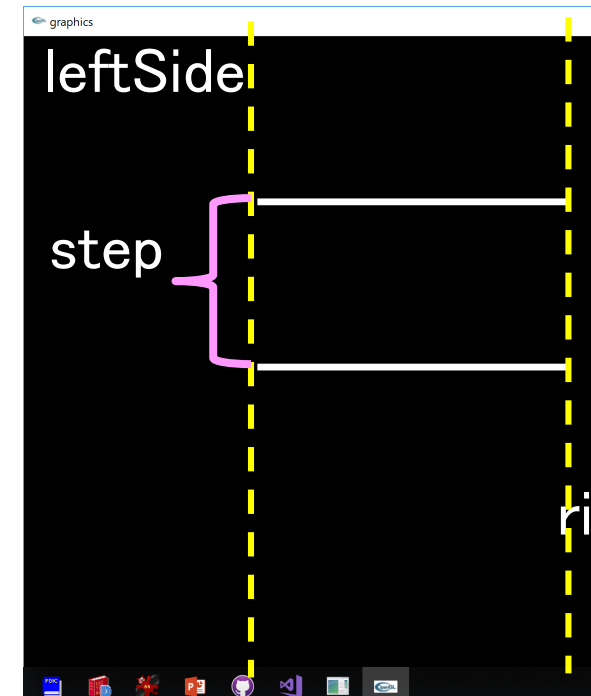


例：隣との距離を表す変数

```
double leftSide  = -40.0; // 左端
double rightSide = 40.0;  // 右端
double step      = 40.0;

glBegin(GL_LINES);
glVertex3d(leftSide, step * 0, 0);
glVertex3d(rightSide, step * 0, 0);

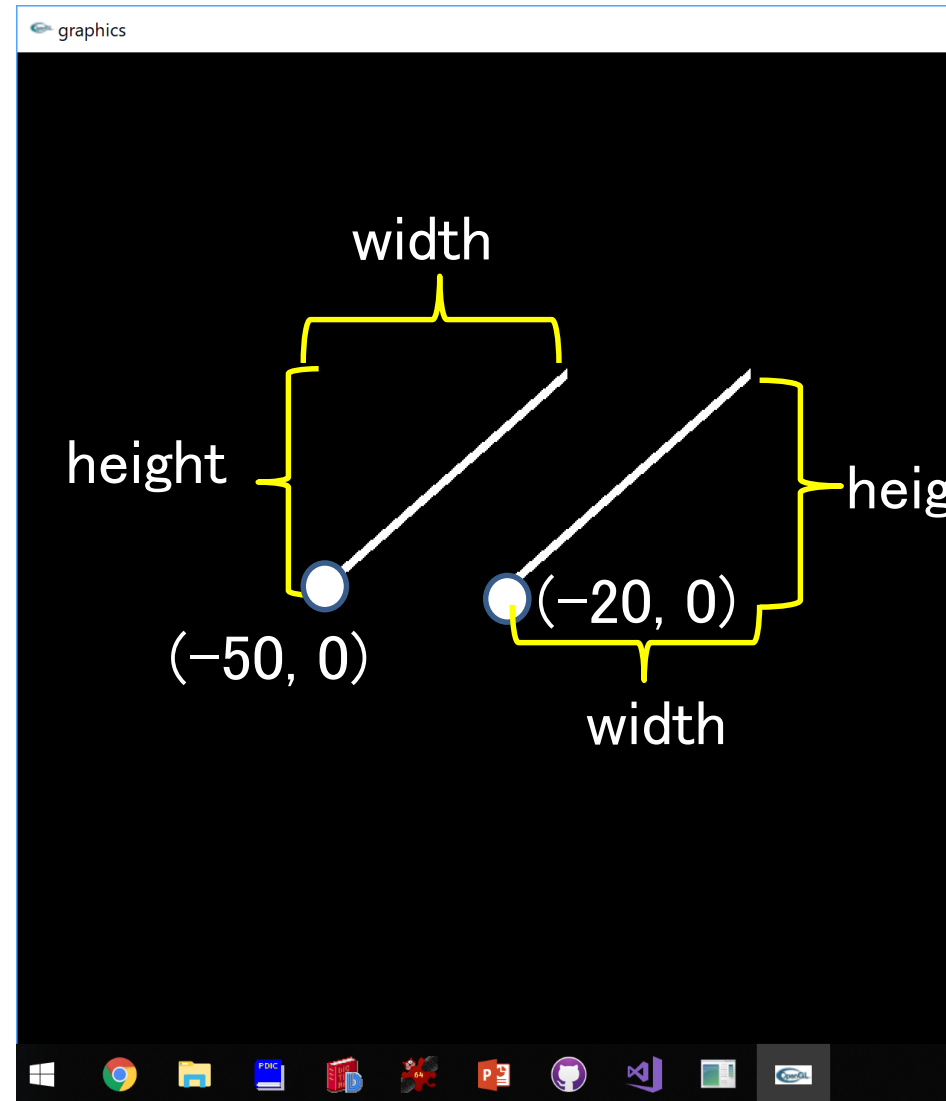
glVertex3d(leftSide, step * 1, 0);
glVertex3d(rightSide, step * 1, 0);
glEnd();
```



例：同じ長さ・方向を表す変数

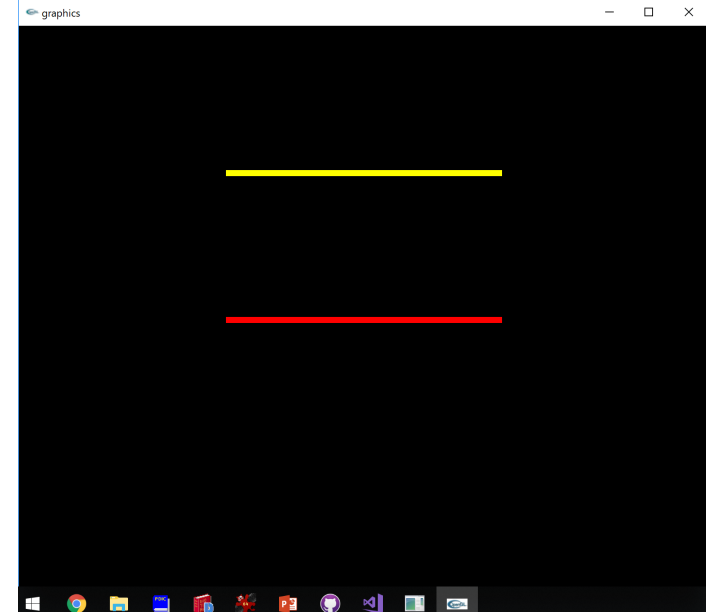
```
double width = 40.0;
double height = 40.0;
glBegin(GL_LINES);
glVertex3d(-20, 0, 0);
glVertex3d(-20 + width,
           0 + height, 0);

glVertex3d(-50, 0, 0);
glVertex3d(-50 + width,
           0 + height, 0);
glEnd();
```



例：色を表す配列

```
double red[3] = {1.0, 0.0, 0.0};  
double yellow[3] = {1.0, 1.0, 0.0};  
glBegin(GL_LINES);  
glColor3d(red[0], red[1], red[2]);  
glVertex3d(-40.0, 0, 0);  
glVertex3d(40.0, 0, 0);  
  
glColor3d(yellow[0], yellow[1],  
          yellow[2]);  
glVertex3d(-40.0, 50, 0);  
glVertex3d( 40.0, 50, 0);  
glEnd();
```



演習課題BASIC

- 線を駆使した見栄えのする映像を制作
 - **制約事項**(次ページ)を守っていれば何でもアリ
 - 題材は自由 図形を描いてもOK
 - 線の種類(直線、曲線、折れ線など)、長さ、方向、太さは全て自由
 - モノクロでもカラーでもOK
 - 静止画像でもアニメーションでもOK
 - プログラムの書き方も自由
 - for文やif文を使ってもOK(来週以降の講義内容...)

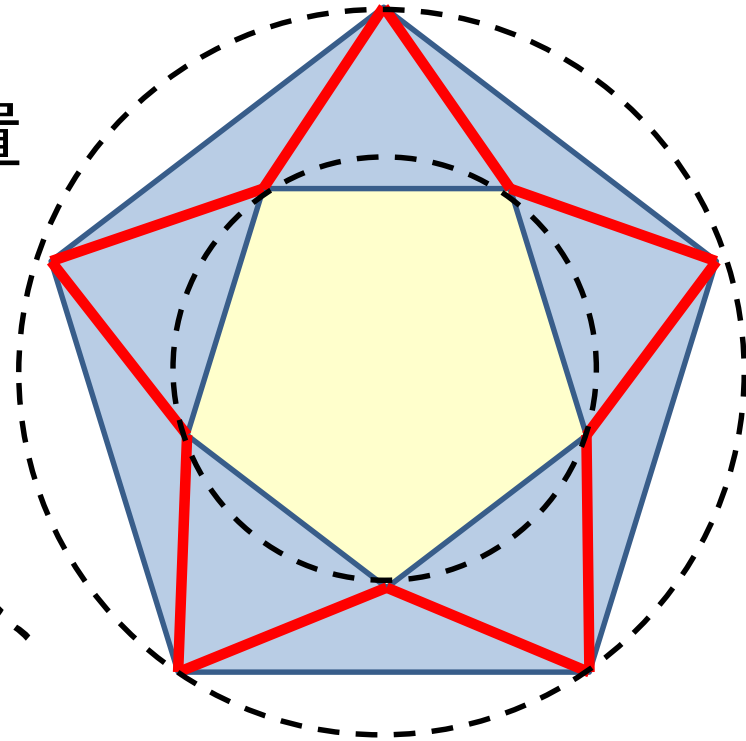
制約事項

1. 線の本数は6本以上(上限ナシ)
2. 1つの変数あるいは配列を編集するだけで、複数の線を同時に操作できるようにしておく
 - 1つの変数を書き換えるだけで、全ての線の太さを一括で操作する とか
 - グループ分けされた図形の大きさを、それぞれのグループの大きさを表す変数・配列を通じて一括操作する や
 - 配列の値を書き換えるだけで、いくつかの線の色を一括して変更する など
3. 用意した変数の機能をプログラム中にコメント
 - スラッシュ2つ「//」以降はコメントとみなされる

演習課題EXTRA

数式を用いて星型の線を描画

- 星を構成する10個の頂点位置を計算(右図参照)
- 星の形と大きさを操作するための変数を用意
 - hint: 2つの外接円の半径
 - ad-hint: 外接円の中心位置ズレ、五角形の回転量を指定
 - さらにはアニメーションも？



提出方法

- graphics/main.cpp を修正しコミット & プルリク
– ファイル名を変える必要はありません
 - 上書き前の内容は全てGithubに保存されています
- 不安な人は、オリジナルの main.cpp を別名でバックアップしておく