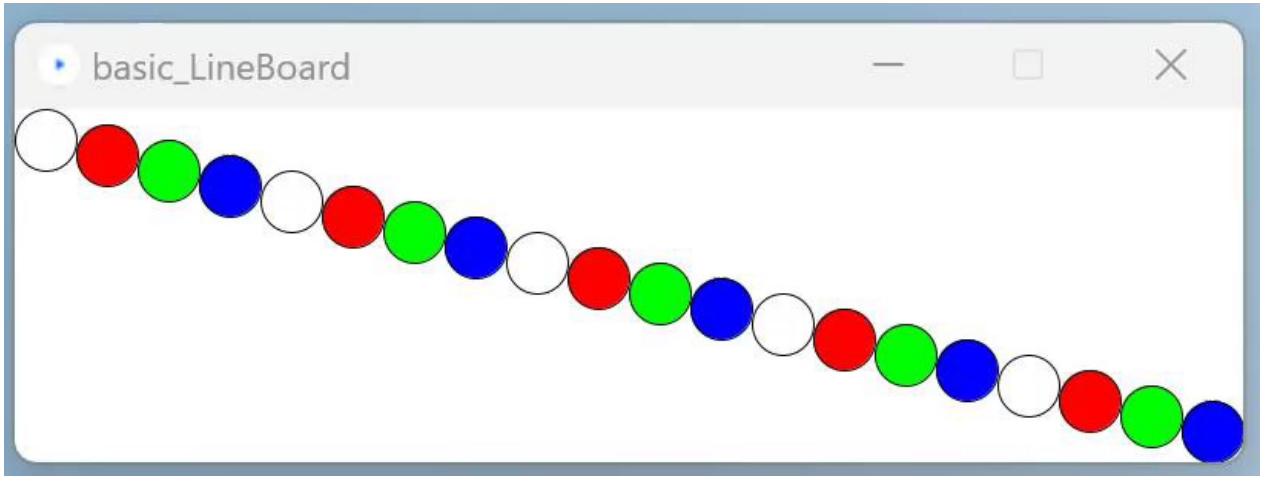
プログラミング演習I （第8回） 課題

- **基本① basic_LineBoard**
  - 800x230のウインドウを作成し、直径40の円を左上から右下にかけて斜めに等間隔に20個並べよ（円の中心のY座標は20から210まで変化させよ）。また円の内部をクリックする度に、そのクリックされた円の色が【白→赤→緑→青→白（以後ループ）】と変化させるようにせよ。
  - なお、起動したときの円の色は下記のようになるようにせよ。



動画：https://gyazo.com/b614cbc3410d53bf16c2c7a52fa407ed

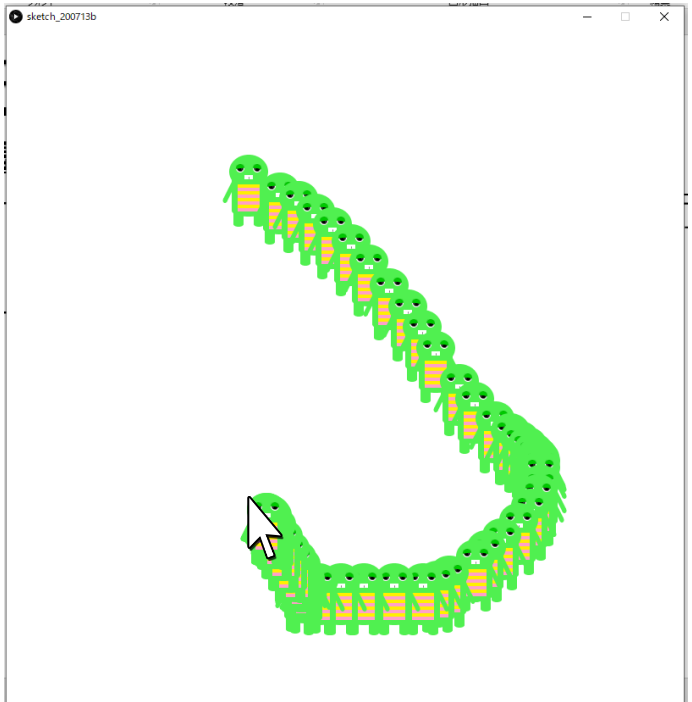
プログラミング演習I（第8回） 課題

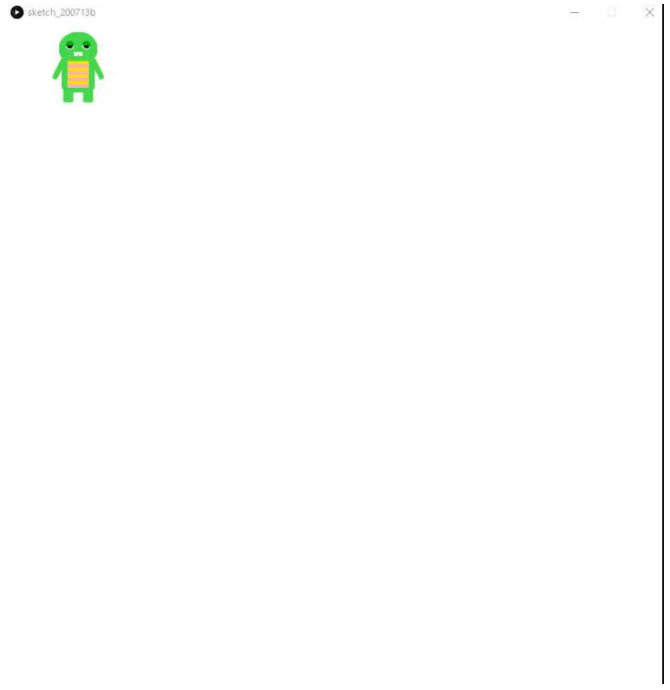
- **基本②　basic_AvgMaxMin**
  - 起動時に145cmから200cmまでの10人の身長をランダムに取得し、その値をまずは「#1: 173.24161cm」のように、半角シャープ、1から始まる数字、半角コロン、半角スペース、身長の値、cmという形式で表示せよ。
  - 次に、その中の最大値、最小値、平均の値を求め、右図のように標準出力せよ。
    - 「cm」表示を忘れないように！

```
#1: 173.24161cm
#2: 180.37756cm
#3: 173.16216cm
#4: 182.3948cm
#5: 177.75885cm
#6: 194.30447cm
#7: 148.29167cm
#8: 171.66035cm
#9: 151.28186cm
#10: 153.73384cm
Max is 194.30447cm
Min is 148.29167cm
Avg is 170.62073cm
```

プログラミング演習I （第8回） 課題

- **基本③ basic_MouseTrace**
  - 800x800のウインドウを作成し，そのウインドウの中でマウスカーソルを動かすと，マウスカーソルを追尾する50個のキャラクタを描画せよ
  - ただし，マウスを追尾しているキャラクタのうち，マウスに近いもの（新しいもの）を，手前に表示するようにせよ
  - 最初に左上にキャラクタが集まっててもよい





キャラクタを小さくする

- drawCharacterを下記のように用意しよう

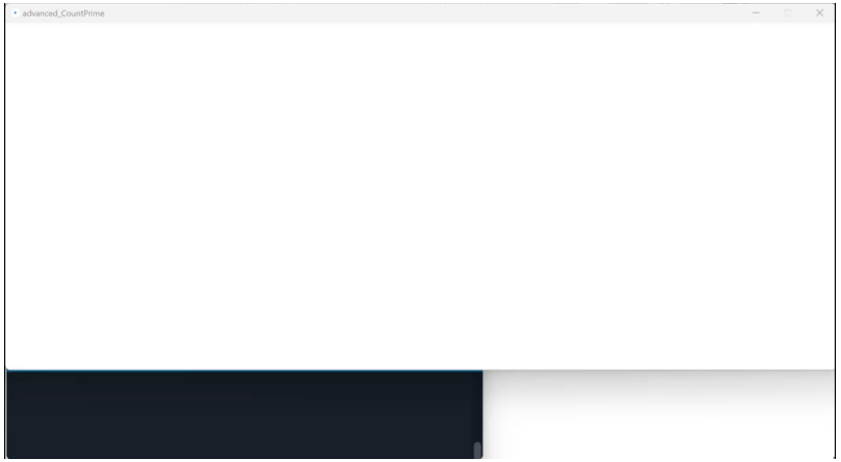
```
void drawCharacter(int cx, int cy)
{
  // scaleで使うための準備。以下の場合は0.2倍にする
  float fScale = 0.2;
  pushMatrix();
  translate(cx-250*fScale, cy-250*fScale);
  scale(fScale);
  // ここから

  ここにキャラクタのプログラムコピペ！


  // ここまで
  popMatrix();
}
```

プログラミング演習I （第8回） 課題

- **発展課題① advanced_CountPrime**
  - 1200x500のウインドウ上に、素数の1の位についてどの数字のものが多いのかを調べ表示するプログラムを作りたい。
  - drawのたびに数を3から順にカウントアップし、その数が素数だったら、その素数を標準出力せよ。また、その値の1の位の値を求めよ。次に、その求めた1の位の値を加算し、棒グラフで数を可視化することによって示せ（数も合わせて提示せよ）。
  - また、そのタイミングでもっとも数が多い数字の棒グラフに色をつけ目立たせるようにせよ



動画：https://gyazo.com/fd02fd07ed69bf1ea38367a93862b4f3

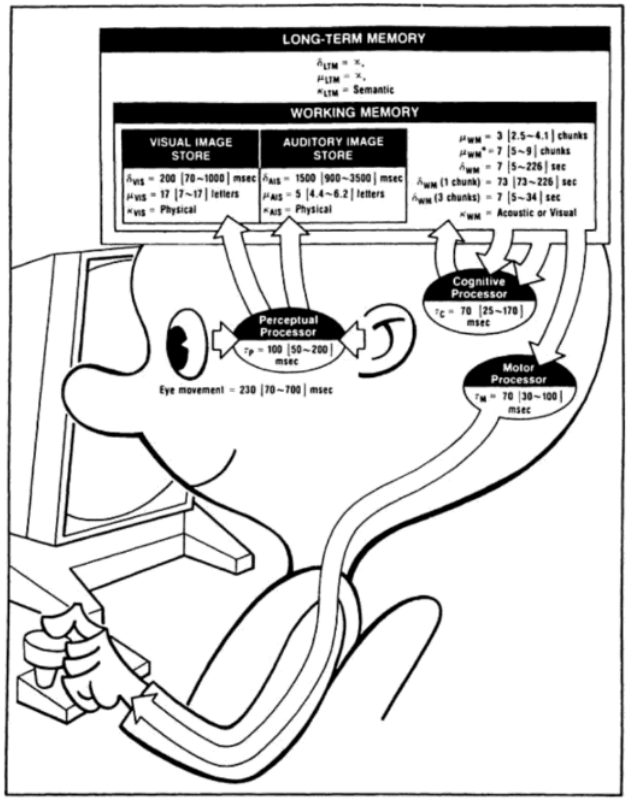
ネタ元：https://twitter.com/HomeiMiyashita/status/1271726234766962688

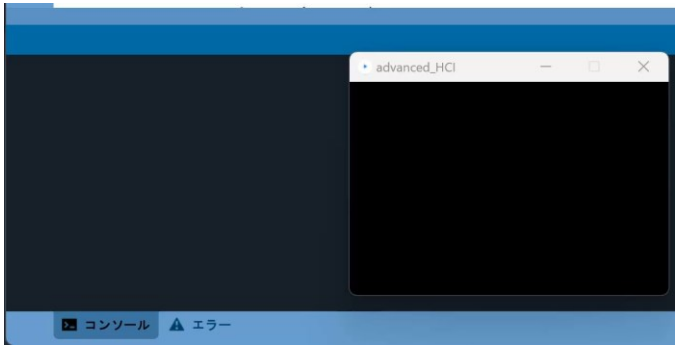
ヒント

- drawのたびに素数かどうかを判定するプログラムを書き、素数ならその数を標準出力
- 素数の場合にその数の1桁の値を求めて、その数を数える
  - 10で割ったあまりが1桁の値だよ！
- 最も数が多いというのは、最大値を求めればできるよね！

プログラミング演習I （第8回） 課題

- **発展②スケッチ名：advanced_HCI**
  - ランプが光ったことに反応してボタンを押すまでの時間は概ね240ミリ秒かかることが知られている［Card 1983］
  - そこで、まず画面の背景を黒色にし、そこからランダムな待ち時間（100～300フレームの待ち時間）ののち、画面を白色にし、その白色にしてからユーザがクリックするとまた画面の背景を黒色にせよ。
  - また、その画面を白色にしてから、クリックするまでの時間をミリ秒単位で計測し、その時間を標準出力せよ。
  - これを10回実施すると、最後には画面に「Finish」とだけ表示し、最短時間、最長時間、平均時間を計算してコンソールに標準出力するプログラムを作成せよ。





動画：https://gyazo.com/91c2f1f0cc5e50253bedf6672b7ecf4d

今日使うテクニック

millis()でミリ秒単位の経過時間を取得する

- アプリケーションが起動されてからの時間は millis() で取得することが可能なので、処理前と処理後の millis()の差分を求めることで、経過時間を取得することができる

```
int start = millis();

// なんか複雑な処理を色々する
// その処理が終わった

int end = millis();
println("経過時間は" + (end-start) + "ミリ秒");
```

今日使うテクニック

millis()でミリ秒単位の経過時間を取得する

- 色々と処理するときはこんな感じの書き方も

```
int iStartMillis;
boolean bFlagStart = false; // スタートしたかどうかのフラグ
void setup() {
  size(300, 150);
  fill( 0 ); // 文字色を黒色に設定
}
void draw() {
  background( 255 );
  if( bFlagStart ){ // スタートしていたら～
    text( millis()-iStartMillis, 20, 90 ); // 差分で経過時間を表示
  }
}
void mousePressed(){
  bFlagStart = true; // クリックされたらスタートフラグを立てる
  iStartMillis = millis(); // スタートの経過時間をセット
}
```