

# プログラミング演習2について

---

- プログラミング演習1と同じところ
  - Processingをやります
  - 最後に発表会をやります
- プログラミング演習1との違い
  - 予習より復習を重視して下さい
  - 中間試験があります
  - 宿題と小テストがあります
  - グループ制作／発表してもらいます

# 何故，試験を？

---

- プログラミングは基本的に，想定通りに動かすことができるが良い
- 研究室に所属したり，企業に就職してプログラムと関わる場合は，知識も必要
  - 人のプログラムを読み理解することができる
  - プログラミングにまつわる言葉を理解する
  - 頭のなかでプログラミングする
- FMSの学生さんには基本情報技術者試験(国家資格)を軽くクリアして欲しい
  - 基本情報技術者試験にはプログラミングがある

- 情報技術全般に関する基本的な知識・技能をもつ者（情報システム開発プロジェクトにおいて、プログラム設計書を作成し、プログラムの開発を行い、単体テストまでの一連のプロセスを担当しているか、将来、そのような業務を担当する者を含む）

# 情報処理技術者試験

---

- 基本情報技術者試験
- 応用情報技術者試験
- 高度試験
  - ITストラテジスト
  - システムアーキテクト
  - ネットワークスペシャリスト
  - 情報セキュリティスペシャリスト
  - ITサービスマネージャ

# グループ制作／発表

---

- 研究室をベースとしてグループ分け予定  
（例）1つの研究室を2グループに分割するなど
- 他者とプログラムする事の難しさ, 楽しさを知る
- 1人ではできない事を実現する！

# 講義の流れ(あくまで目安)

---

- 1コマ目：
  - 0～20: 小テスト実施 (遅刻したらアウト！)
  - 20～30: 小テストの解説(答え合わせ)
  - 30～50: 今日のトピックの解説＋課題の提示
  - 50～90: 今日の課題に取り組む(課題は要提出)
- 2コマ目：
  - 0～30: 今日の課題に取り組む(30分に提出)
  - 30～90: 今日の課題の解説(答え合わせ)
  - 宿題の提示
- **小テストは前週の講義, 課題, 宿題から!**

# 評価

---

- 学期始めテスト+小テスト(中間まで): 20点
- 中間テスト: 40点
- 授業中課題: 10点(提出)
- 発表会: 30点
  
- 注意
  - 5回休み=単位不認定
  - 課題については演習中にチェックはしない

# 要注意！

---

- 他者のプログラムをコピーしない&コピーさせないこと
- 春学期において他者のプログラムのコピーを発見し、点数減点しています
- 春学期については、事前に厳しく言っていなかったため減点扱いでしたが、今後発見した場合は、問答無用でその学期の全ての単位剥奪となります
- コピーさせた人も同様の処分になることに注意してください！

# 講義予定(案)

---

- 第1回 [9/21] ガイダンス＋前期の復習
- 第2回 [9/28] クラス (1)
- 第3回 [10/5] クラス (2)
- 第4回 [10/12] 継承
- 第5回 [10/19] クラス＋継承(復習)
- 第6回 [10/26] ArrayList
- 第7回 [11/9] 中間テスト
- 第8回 [11/16] 振り返りとファイル入出力
- 第9回 [11/23] ネットワーク
- 第10回 [11/30] Web API
- 第11回 [12/7] フィジカルコンピューティング
- 第12回 [12/14] フィジカルコンピューティング
- 第13回 [12/21] フィジカルコンピューティング
- 第14回 [1/18] 発表会(18時～ホールで実施予定)

# 復習1-1, 1-2

- 出力結果はどうなるか？

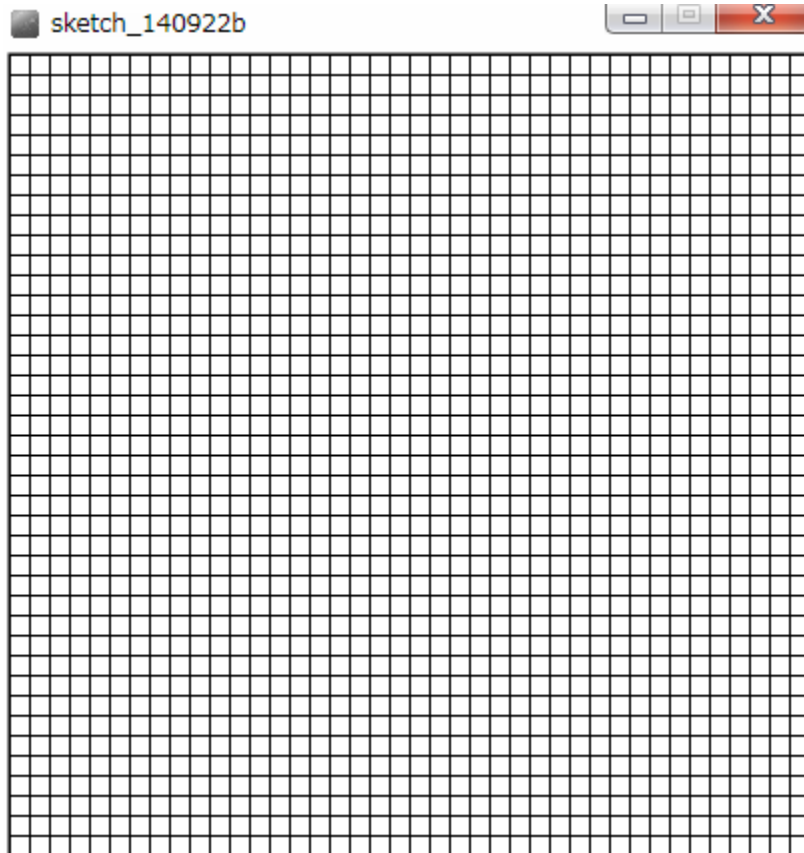
```
1 int x, y, z; ←
2 x = 10; ←
3 y = 20; ←
4 println( x + y ); ←
5 z = x + y; ←
6 z = z * 5; ←
7 println( z ); ←
8 println( x / z ); ←
9 ←
```

- ウィンドウの右半分にカーソルがある場合に背景の色を赤色にし, 左半分にカーソルがある場合に背景の色を白色にするには？

```
1 void setup() { ←
2   ^ size( 400, 400 ); ←
3 } ←
4 ←
5 void draw() { ←
6   ^ if( A ) { ←
7     ^ background( 255, 255, 255 ); ←
8     ^ } else { ←
9     ^ background( B ); ←
10    ^ } ←
11 } ←
```

# 復習1-3(繰り返し)

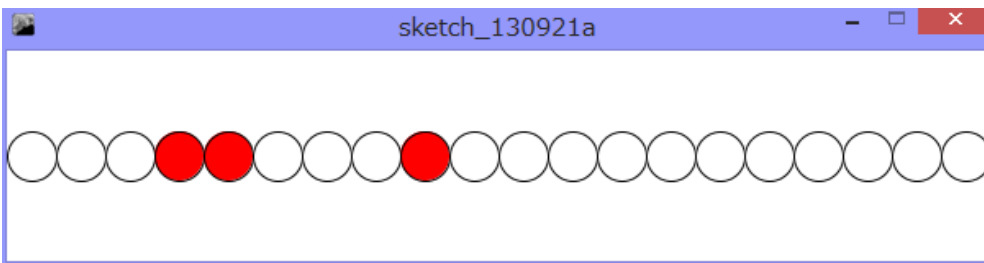
- プログラムの問題を指摘せよ
  - 左側のような表示を行いたいが、動作しない。問題を指摘せよ。



```
1 size( 400, 400 );  
2 background( 255 );  
3 int i = 0;  
4 while( i < 400 ){  
5 ^   i = 0;  
6 ^   line( i, 0, i, 400 );  
7 ^   i = i + 10;  
8 }  
9 while( i < 400 ){  
10 ^   i = 0;  
11 ^   line( 0, i, 400, i );  
12 ^   i = i + 10;  
13 }  
14
```

# 復習1-4(配列)

- 下記のような表示のプログラムを作るにはどうしたらよいか、右記プログラムの穴をうめよ



```
int [] light = new int [20];
void setup(){
  size( 600, 100 );
  for( int x=0; x<20; x++ ){
    light[x] = 0;
  }
  light[ A ] = 1;
  light[ B ] = 1;
  light[ C ] = 1;
}
```

```
void draw(){
  background( 255, 255, 255 );
  int x = 0;
  while( D ){
    if( light[x] == 1 ){
      fill( 255, 0, 0 );
    } else {
      fill( 255, 255, 255 );
    }
    ellipse( E, 50, 30, 30 );
    x++;
  }
}
```

# 復習1-5(関数／メソッド)

- 整数の値を引数として渡すと、約数の数を返す関数を作成せよ

```
int getNumberOfDivison( int num ){
    int i=1;
    int count = A ;
    while( i<=num ){
        if( B ){
            count++;
        }
        i++;
    }
    return C ;
}
```

# 課題1-1

---

- 1から200までカウントアップしながら、状況に応じて文字を表示するプログラムを作成せよ
    - 3で割り切れる場合は「F」
    - 5で割り切れる場合は「M」
    - 7で割り切れる場合は「S」
    - 3と5で割り切れる場合は「FM」
    - 5と7で割り切れる場合は「MS」
    - 3と7で割り切れる場合は「FS」
    - 3と5と7で割り切れる場合は「FMS」
- を数の代わりに出力するというもの
- 下記のように1～200までについて、連続的に表示するようにせよ(1つずつ改行してもよい)

## 【出力例】

1, 2, F, 4, M, 6, S, 8, F, M, 11, F, 13, S, FM, ...

# 宿題1-1

---

- 復習1-5で作成した約数の数を返す関数を活用し、引数として指定した数字が素数かどうかを判定する関数(素数判定関数)を作成せよ
  - (ヒント) 約数の数が2だったら1を返し、2出なかったら0を返す関数を作る
- その関数(メソッド)を利用して、2から各自の学生番号(8桁)までのすべての整数について素数かどうかを判定し、素数の場合はその値をprintlnで標準出力するプログラムを作成せよ

# 宿題1-2

- コラッツ予想とは、下記のルールに従うとすべての自然数が最終的に1になるのではという予想である。まだ、本予想は証明されていないが、 $3 \times 2^{53}$ までは反例が無いことが確かめられている。ルールは下記のとおり。
  - ある数が偶数なら2で割る
  - ある数が奇数なら3を掛けて1を足す
  - 計算結果が1になるまで上記の計算を繰り返す
- ここで、2から100までの数字において、すべての値の変化を右下のように表示せよ。また、その最後にステップ数も表示せよ。なお、そのために下記の仕様を満たす関数Collatzを作成せよ
  - $n$ が偶数の時  $Collatz(n) = \frac{n}{2}$
  - $n$ が奇数の時  $Collatz(n) = 3n + 1$

```
[2]->1 (1 step)
[3]->10->5->16->8->4->2->1 (7 steps)
[4]->2->1 (2 steps)
[5]->16->8->4->2->1 (5 steps)
[6]->3->10->5->16->8->4->2->1 (8 steps)
[7]->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (14 steps)
[8]->4->2->1 (3 steps)
[9]->28->14->7->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (16 steps)
[10]->5->16->8->4->2->1 (6 steps)
[11]->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (13 steps)
[12]->6->3->10->5->16->8->4->2->1 (9 steps)
[13]->40->20->10->5->16->8->4->2->1 (9 steps)
[14]->7->22->11->34->17->52->26->13->40->20->10->5->16->8->4->2->1 (15 steps)
[15]->46->23->70->35->106->53->160->80->40->20->10->5->16->8->4->2->1 (17 steps)
[16]->8->4->2->1 (4 steps)
[17]->52->26->13->40->20->10->5->16->8->4->2->1 (12 steps)
```