



プログラミング演習(7)

多重の繰り返し

中村, 高橋
小林, 橋本

九九の表を作る



標準出力で九九の表を作る！

考え方

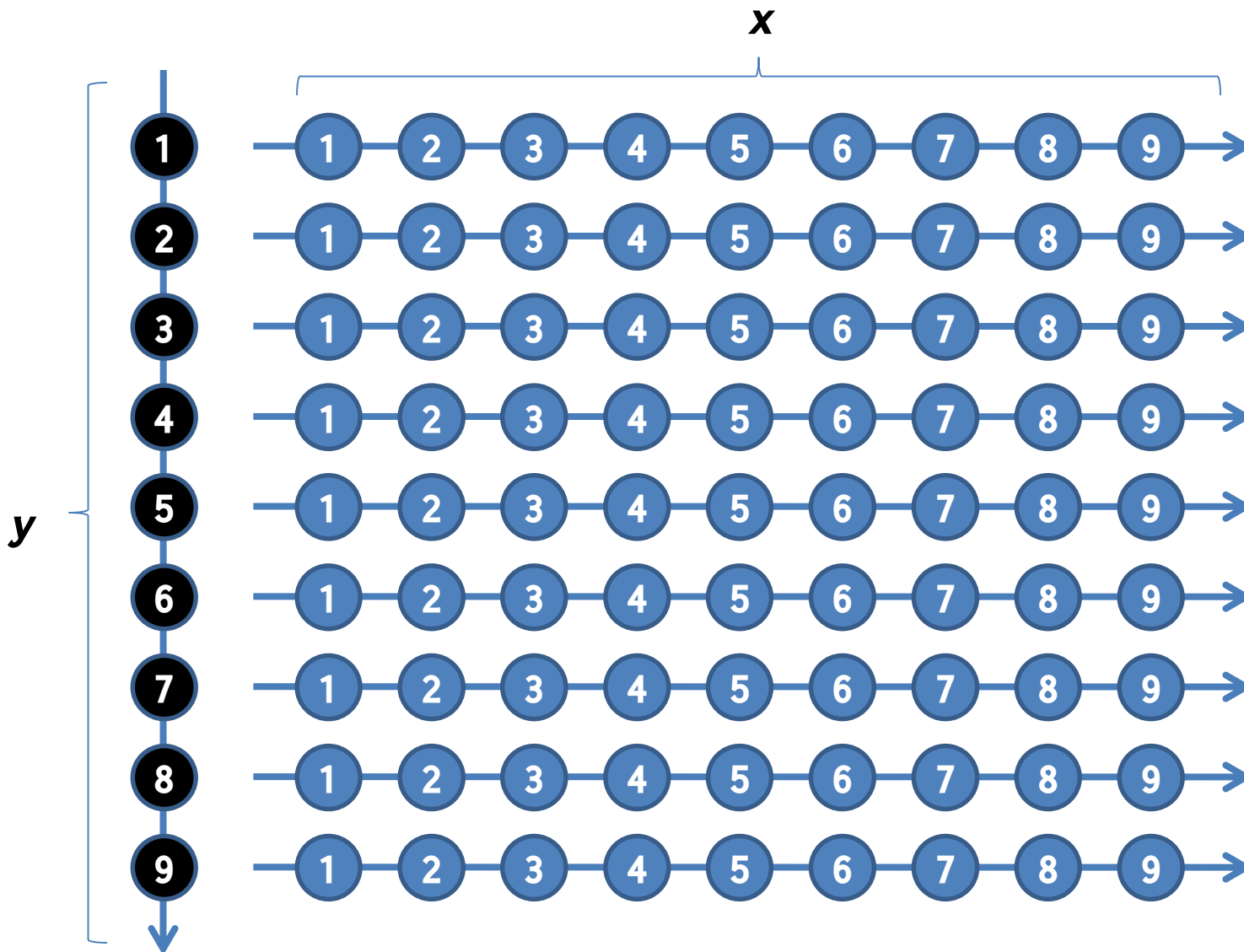
- 行と列をそれぞれの繰り返しに分離する
- 計算結果をそのまま表示する

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

二重ループ

九九の表： 多重ループ

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室





• 考え方

- 九九の計算は $1*1, 1*2, 1*3, \dots, 9*7, 9*8, 9*9$ のような1~9までの数字の掛け算の組み合わせ
- それぞれ変数を x と y とし, 1から9まで変化させる
- x を1から9まで1ずつ増やし, x が10になると y を増やして, x を1にする
- 掛け算の結果は $x*y$ で表示できる
- 適当にスペースを入れていく



- 答えの例

- 桁そろえのためにスペースを入れるとなおよし

```
int y = 1;

while(y <= 9)
{
    int x = 1;
    while(x <= 9)
    {
        print(" ");
        print(x * y);
        x++;
    }
    y++;
    println();
}
```

```
for(int y = 1; y <= 9; y++)
{
    for(int x = 1; x <= 9; x++)
    {
        print(" ");
        print(x * y);
    }
    println();
}
```

九九の表（桁そろえ）



```
int y = 1;

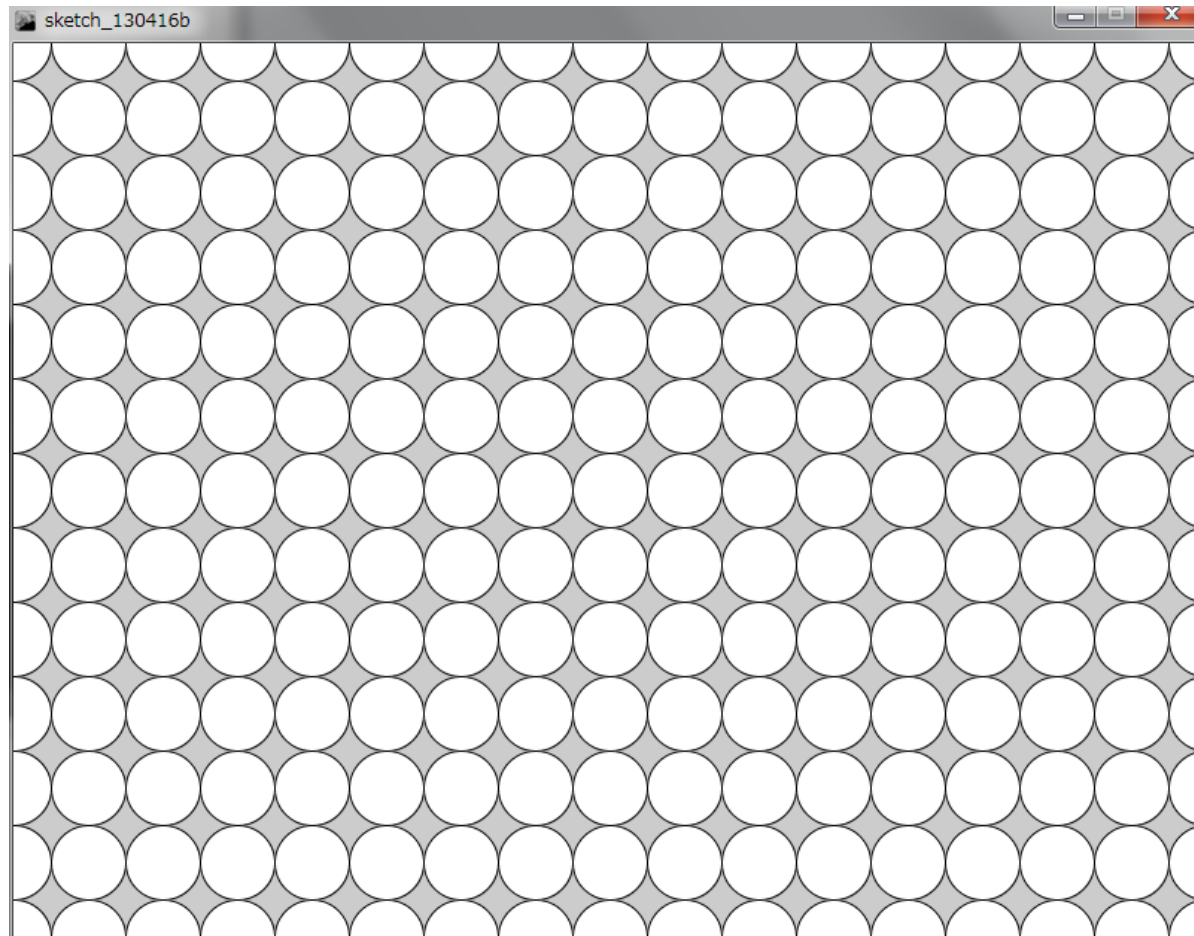
while(y <= 9)
{
    int x = 1;
    while(x <= 9)
    {
        print(" ");
        if(x * y < 10)
        {
            print(" ");
        }
        print(x * y);
        x++;
    }
    y++;
    println();
}
```

```
for(int y = 1; y <= 9; y++)
{
    for(int x = 1; x <= 9; x++)
    {
        print(" ");
        if(x * y < 10)
        {
            print(" ");
        }
        print(x * y);
    }
    println();
}
```

円を敷き詰めてみる



(Q) 800x600のウィンドウに直径50の円を敷き詰めるにはどうしたら良いか？



円を敷き詰めてみる



- 考え方

- 円の中心座標を x, y とする
- ある y のときに, 円の中心のX座標は $x, x+50, x+100, x+150, x+200, \dots$ となる
- x は繰り返しで50ずつ増やし, 400まで来ると下の段へと移動したい
- 下の段への移動するには, y を50増やす
- y を50増やすときに x を 0 にする

円を敷き詰めてみる



```
size(800, 600);  
int x = 0;  
int y = 0;
```

円の中心の座標を
変数x, yとし, その値を0にする

```
while(y <= height)
```

y<=heightまで繰り返す

```
{
```

```
  while(x <= width)
```

x<=widthまで繰り返す

```
{
```

```
  ellipse(x, y, 50, 50);
```

x を 50 ずつ増やす

```
  x = x + 50;
```

```
}
```

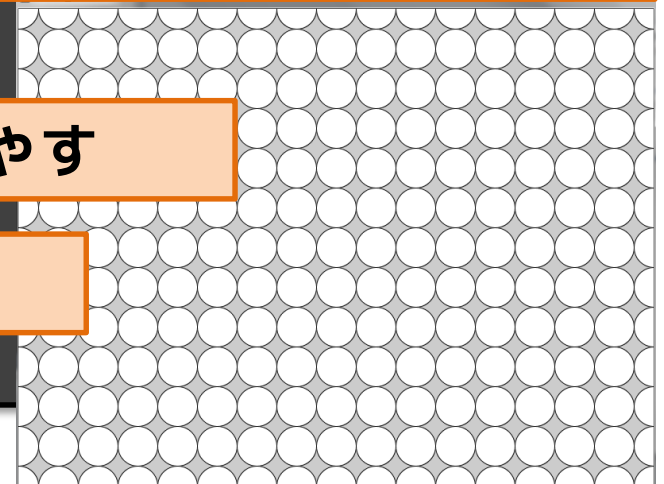
y を 50 ずつ増やす

```
y = y + 50;
```

```
x = 0;
```

x を 0 に戻す

```
}
```



円を敷き詰めてみる



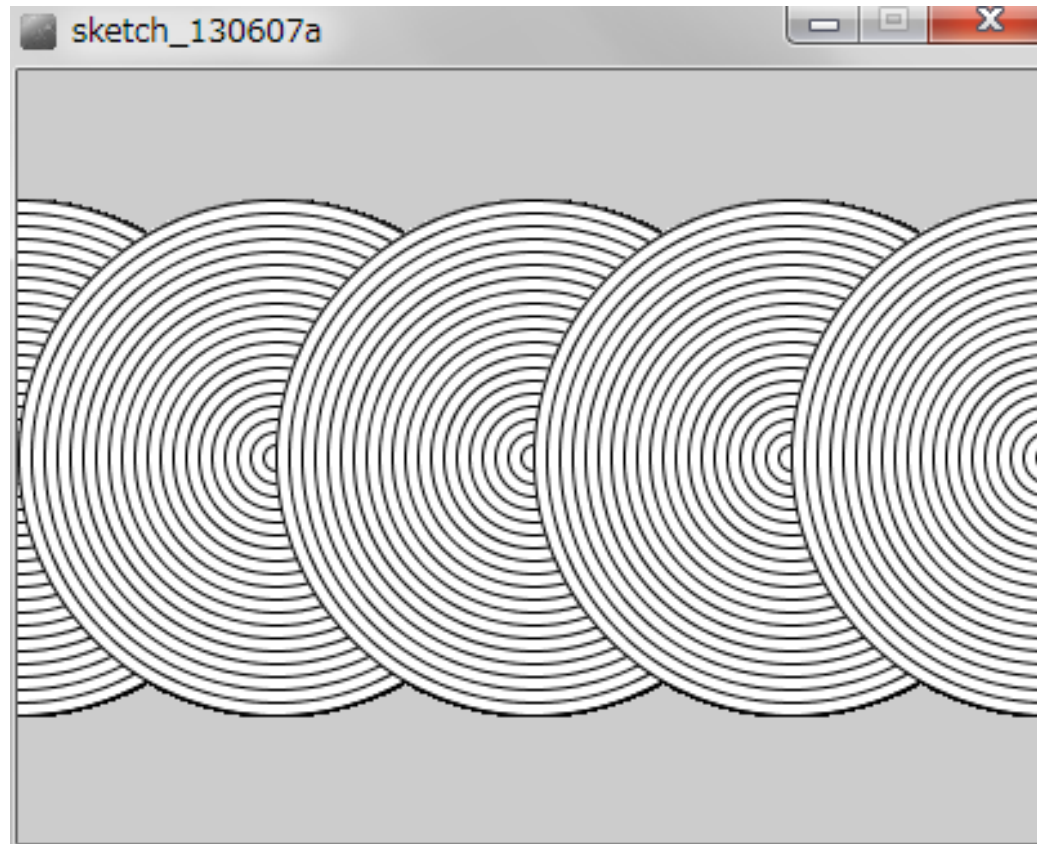
- 多重ループの場合，初期化がややこしくなるので for を使うとよりシンプルになる

```
size(800, 600);  
  
for(int y = 0; y <= height; y += 50)  
{  
    for(int x = 0; x <= width; x += 50)  
    {  
        ellipse(x, y, 50, 50);  
    }  
}
```

沢山の円の中に円を描く



(Q) 400x300のウィンドウに，下図のように図形を描くにはどうするか？中心の座標は100ずつずれており最大の半径は100で，5ずつ小さくなる



沢山の円の中に円を描く



• 考え方

- 最も大きな円の中心座標は, $0, 100, 200, 300, 400, 500$ と100ずつ増えていく
- 最も大きな円の半径は100で, 内部の円の半径は $95, 90, 85, \dots, 10, 5$ と小さくなっていく
- 2つのループの外側で大きな円の中心座標を, 内側で半径の大きさを変化させていく
- 円の中心座標 $x = 0, 100, 200, 300, 400, 500$
- 円の半径 $r = 95, 90, 85, \dots, 10, 5$

沢山の円の中に円を描く



- 注意点： 円の半径を徐々に大きくしていくと最後の大きな円しか描画されなくなる！

```
size(400, 300);
int x = 0;
while (x < 5)
{
    int r = 100;
    while (r > 0)
    {
        ellipse(x * 100, 150, r * 2, r * 2);
        r = r - 5;
    }
    x++;
}
```

沢山の円の中に円を描く



- 注意点： 円の半径を徐々に大きくしていくと最後の大きな円しか描画されなくなる！

```
size(400, 300);

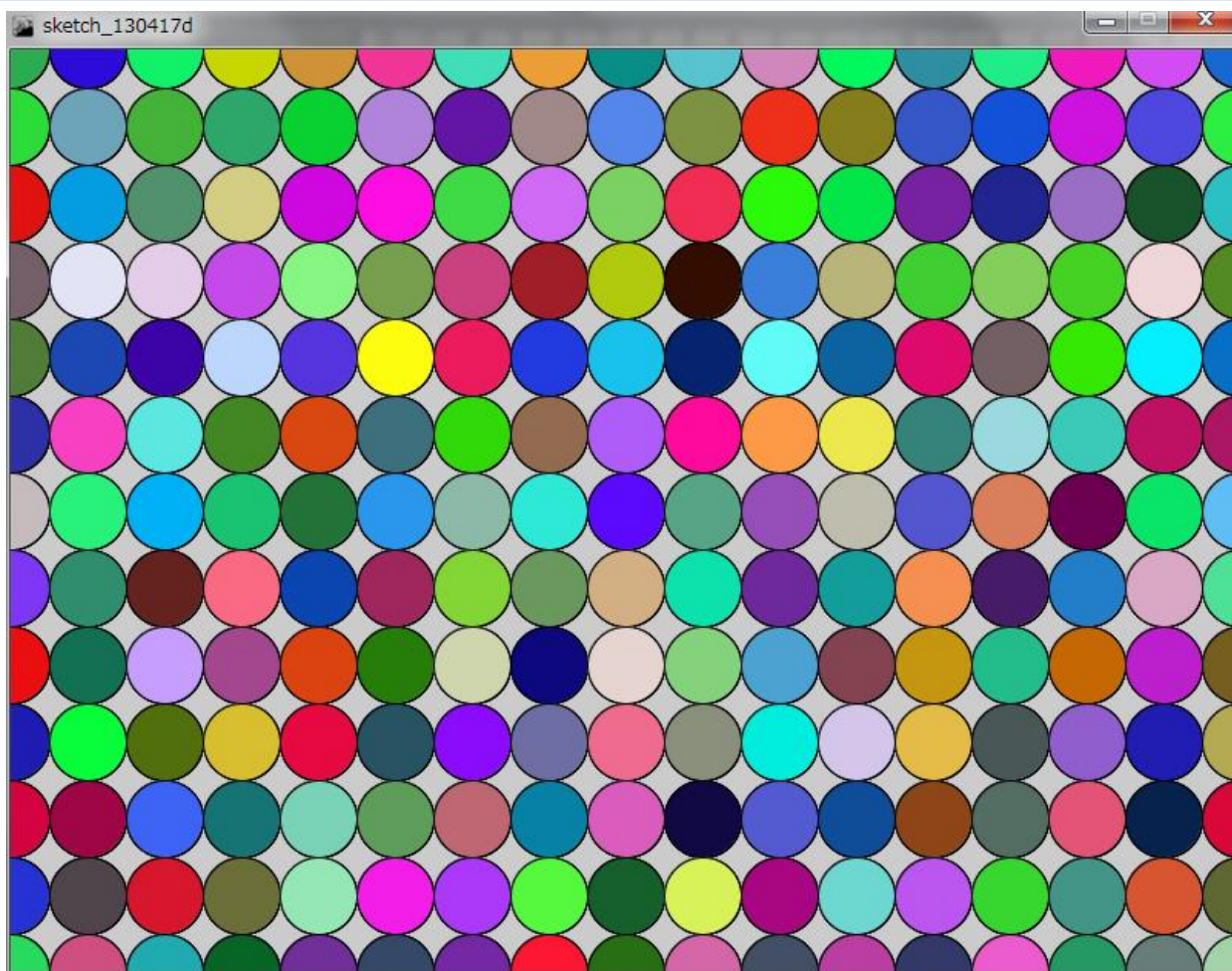
for(int x = 0; x < 5; x++)
{
  for(int r = 100; r > 0; r -= 5)
  {
    ellipse(x * 100, 150, r * 2, r * 2);
  }
}
```

敷き詰めた円の色を変えよう

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



(Q) 800x600のウィンドウに敷き詰めた円の色を変えてみよう



敷き詰めた円の色を変えよう



- 敷き詰めた円の色を乱数で指定
- 0~255までの乱数はrandom(256)で取得できる

```
size(800, 600);

for(int y = 0; y <= width; y = y + 50)
{
  for(int x = 0; x <= height; x = x + 50)
  {
    fill(random(256), random(256), random(256));
    ellipse(x, y, 50, 50);
  }
}
```


画面を数字で埋め尽くす



(Q) なるべく画面いっぱいにできるだけ数字を敷き詰めてみましょう。また、その数字はランダムに変わるようにせよ

考え方

- ランダムな0~9までの数字は $\text{random}(0, 10)$ で求めることができる
- $\text{random}(0, 10)$ で取得できる値は実数なので、整数 (int) に変換する



```
void setup()
{
  size(1280, 768); // サイズに応じて
}
void draw()
{
  background(255); // 背景は白色に
  textSize(50); // 文字サイズを50にする
  fill(0); // 文字は黒色で
  int x = 0;
  int y = 0;
  // x * 50 が右端に来るまで繰り返す
  while (x * 50 < width)
  {
    y = 0; // 繰り返しの前に y を初期化しておく
    while (y * 50 < height)
    { // y*50 が下端に来るまで繰り返す
      int num = (int)random(0, 10); // 0~9までの乱数取得
      text(num, x * 50, y * 50); // 取得した値を(x * 50, y * 50)に表示
      y = y + 1;
    }
    x = x + 1;
  }
}
```



```
void setup()
{
  size(1280, 768); // サイズに応じて
}

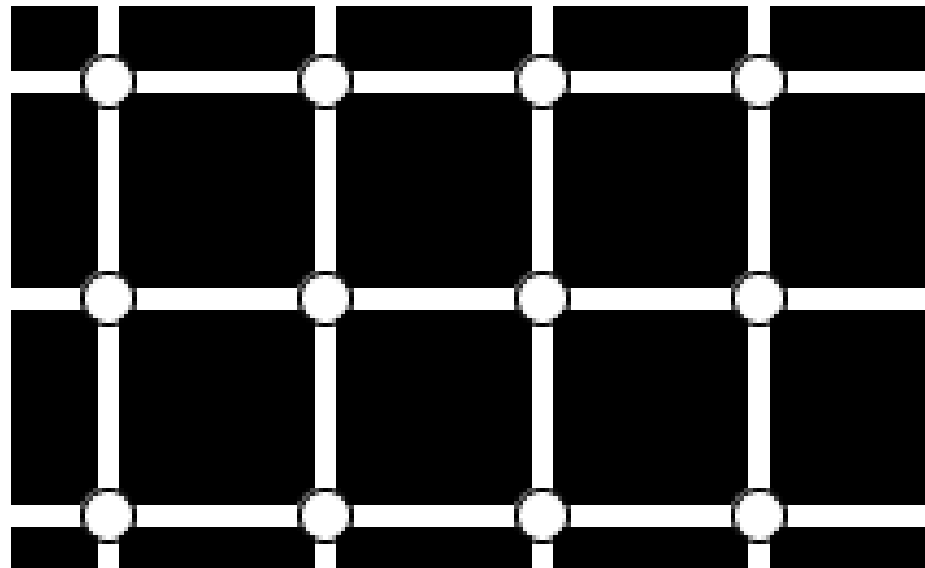
void draw()
{
  background(255); // 背景は白色に
  textSize(50); // 文字サイズを50にする
  fill(0); // 文字は黒色で

  // x*50 が右端に来るまで繰り返す
  for(int x = 0; x * 50 < width; x++)
  {
    // y*50 が下端に来るまで繰り返す
    for(int y = 0; y * 50 < height; y++)
    {
      int num = (int)random(0, 10); // 0~9までの乱数取得
      text(num, x * 50, y * 50); // 取得した値を(x*50,y*50)に表示
    }
  }
}
```

錯覚を作ってみよう！



(Q) 800×800 のウィンドウを白色で塗りつぶし, 黒い正方形(1辺44ピクセル)を, 上下左右に6ピクセルずつあけながら敷き詰める. さらに, 敷き詰められた正方形によって作られた上下の白い道の交点に直径14ピクセルの白色の円を描く!



錯覚を作ってみよう！



- 考え方

- 2つのループを作って黒い正方形を敷き詰める

- 上下左右に6ピクセルずつあけると言うことは，1つの図形を描画する際には3ピクセルずつずらせばよい！
 - 基本となる位置を(x, y)とすると，`rect(x, y, 44, 44);`
 - 1回の繰り返し毎に50

ピクセルずつxやyを
増やしていく！



6ピクセル 44ピクセル 6ピクセル

錯覚を作ってみよう！



• 考え方

- ある黒い正方形の左上の座標を (x, y) としたとき, 正方形によって作られた格子の交点の中心座標は $(x-3, y-3)$ となる
- 交点の中心に直径14ピクセルの白い円を描く
 - `ellipse(x-3, y-3, 14, 14);`
- x と y を2つの繰り返しを使い50ずつ増やしなから, 右端, 下端まで円を描画する

whileでやってみる



```
size(800, 800);
background(255);
int x = 0;

while(x <= width)
{
  int y = 0;
  while(y <= height)
  {
    noStroke();
    fill(0);
    rect(x, y, 44, 44);
    y = y + 50;
  }
  x = x + 50;
}
```

```
int x = 0;

while (x <= width)
{
  int y = 0;
  while (y <= height)
  {
    stroke(0);
    fill(255);
    ellipse(x - 3, y - 3, 14, 14);
    y = y + 50;
  }
  x = x + 50;
}
```

forを

```
size(800, 800);  
background(255);
```

```
for(int x = 0; x <= width; x = x + 50)  
{  
  for(int y = 0; y < height; y = y + 50)  
  {  
    noStroke();  
    fill(0);  
    rect(x, y, 44, 44);  
  }  
}
```

```
for(int x = 0; x <= width; x = x + 50)  
{  
  for(int y = 0; y < height; y = y + 50)  
  {  
    stroke(0);  
    fill(255);  
    ellipse(x - 3, y - 3, 14, 14);  
  }  
}
```

学科



素数を求めよう



(Q) 1000までの全ての素数を求める

• 考え方

- 素数とは、その値と1以外に正の約数がないもの
- つまり、正の約数の数が2だったら素数である！
 - 約数の数の求め方は前回の資料を参照
 - 2から999までに正の約数がない場合は、素数！
- 調べたい数字 `num` を2から999まで変化させて、その数毎に正の約数の数を数え、約数が0個だったらこの数は素数と表示！



```
int num = 1;
while(num <= 1000)
{
    int i = 2;
    int count = 0;
    while(i < num)
    {
        if((num % i) == 0)
        {
            // numをiで割った余りが
            // 0だったらcountを増やす
            count++;
        }
        i++;
    }
    if(count == 0)
    {
        println(num + "は素数です!");
    }
    num++;
}
```

素数を求めよう



```
for(int num = 1; num <= 1000; num++)
{
    int count = 0;
    for(int i = 2; i < num; i++)
    {
        if((num % i) == 0)
        {
            // numをiで割った余りが
            // 0だったらcountを増やす
            count++;
        }
    }
    if(count == 0)
    {
        println(num + "は素数です!");
    }
}
```

素数を求めよう



- ちなみに、ある値 N の約数は \sqrt{N} より大きくはならないので（例えば、 N の二乗根が n のときに、 n より大きな値は約数になりえない）

```
for(int num = 1; num <= 1000; num++)
{
    int flag = 1;
    for(int i = 2; i <= sqrt(num); i++)
    {
        if((num % i) == 0)
        {
            // numをiで割った余りが0だったら
            // flag=1にして終わる
            flag = 0;
            break; // ループを抜ける
        }
    }
    if(flag == 1)
    {
        println(num + "は素数です!");
    }
}
```

コインを1万回10セット



(Q) コインを1万回投げた時に、表と裏になる回数はそれぞれ何回か？について、10セット実施し、その表と裏の回数を表示せよ

• 考え方

- 10000万回コインを投げて表裏の数を数えるというタスクを、10回繰り返してそのたびに結果を表示する

1回目:	表	4914,	裏	5086
2回目:	表	5031,	裏	4969
3回目:	表	4970,	裏	5030
4回目:	表	4961,	裏	5039
5回目:	表	5049,	裏	4951
6回目:	表	4980,	裏	5020
7回目:	表	4970,	裏	5030
8回目:	表	4918,	裏	5082
9回目:	表	5004,	裏	4996
10回目:	表	5003,	裏	4997

コイン

```
for (int j = 1; j <= 10; j++)
{
    // 初期化して, 10000回コインを投げる
    int countHead = 0;
    int countTail = 0;
    for (int i = 0; i < 10000; i++)
    {
        // 0なら表, 1なら裏とする
        if ((int)random(2) == 0)
        {
            countHead++;
        }
        else
        {
            countTail++;
        }
    }
    // 回数と結果を表示
    print(j + "回目: ");
    print("表 " + countHead);
    println(", 裏 " + countTail);
}
```





- 1から10000までの素数を表示するプログラムを作成してみましょう
- 敷き詰められた円が左から右, 上から下にグラデーションされるようにしてみましょう
- 1億までのすべての正の完全数を求めてみよう
 - 完全数は鈴木先生の先端メディアサイエンス概論にも登場
 - ある数の約数の和がその数自体になるもの
 - $6 = 1 + 2 + 3$, $28 = 1 + 2 + 4 + 7 + 14$ など



- 完全数の求め方

- すべての約数を求め，その約数の和がその数自体になるかどうかで判定．ただ凄く遅い

- 奇数の完全数がないとすると，ユークリッドとオイラにより下記が証明されている

$$\text{完全数} = 2^{p-1}(2^p - 1)$$

- を利用するともっと高速化できるよ！
- 桁数を増やす場合は，intじゃなくてlongを使う