コンピュータ基礎演習

第7回

理工学部 情報科学科 隅田 康明

sumida@ip.kyusan-u.ac.jp

大学の自習室等の利用について

- 制限付きで自習室などが利用可能
  - 総合情報基盤センター等のPCを利用可能に
    - https://www.cnc.kyusan-u.ac.jp/aboutus/003560.html
    - 大学のPCにはProcessing、PowerPointがインストール済み（全部ではないですが）
- 通信環境の問題で、授業の受講が困難な場合も、自習室などで遠隔授業を受けることができます
  - https://www.kyusan-u.ac.jp/news/coronavirus20200519-2/
- 自宅で授業を受けられない場合には検討しましょう

  ※大学で作業することを推奨しているわけではありません

講義用HPについて

- 講義資料や、講義に必要な情報は、
  この授業用のホームページに掲載しています
  - K'sLifeの通知にも添付していますが、HPの方がアクセスはしやすいはずです
    - K'sLifeに接続しにくい状況が続いています
  - 念の為、K'sLifeにもアップロードはしますが、
    基本的には講義HPを見るようにしましょう
- 下記のHPから、講義資料、動画のURL、
  Zoomの招待リンクなどを確認できます。

http://www.is.kyusan-u.ac.jp/~sumida/class/pckiso/

  - だんだんMoodleに移行してきます
    - Moodleに行けば全部ある！の方が楽ですよね

遠隔授業期間中の質問

- **困ったら早めに質問・相談！！**
  - 学生側から質問されないと、
    誰が困っているのか、何が分からないのか、分かりません
- メール：やり取りに時間はかかるが一番確実
- Zoom：授業時間中限定
  - 時間は限られるが、作業中の画面を見ながら教えられるので、問題を短時間で解決出来る可能性が高い
- Line OpenChat：授業時間中限定
  - 文字だけのやり取りに限定（画像アップロードは禁止）
  - 質問内容が他の学生にも分かるので注意すること
    - プログラムの全文貼り付け等は厳禁！

授業についての質問メールについて

[授業名(曜日時限)]についての質問	件名
～先生	誰宛か
[授業名(曜日時限)]を受講しています、 20AA999の九産太郎です。	何者か
（質問内容） -- 20AA999 九産太郎 九州産業大学　芸術学部　○○学科 1年	用件 署名

2日返信がなければもう一度送って下さい（なるべく見落としたくない）

各回のレポートの確認（必ず確認しておくこと）

提出ステータス

提出ステータス	評定のために提出済み
評定ステータス	未評定
終了日時	2020年 05月 25日(月曜日) 00:00
残り時間	4 日 2 時間
最終更新日時	2020年 05月 20日(水曜日) 21:19

提出すると「評定のために提出済み」になる

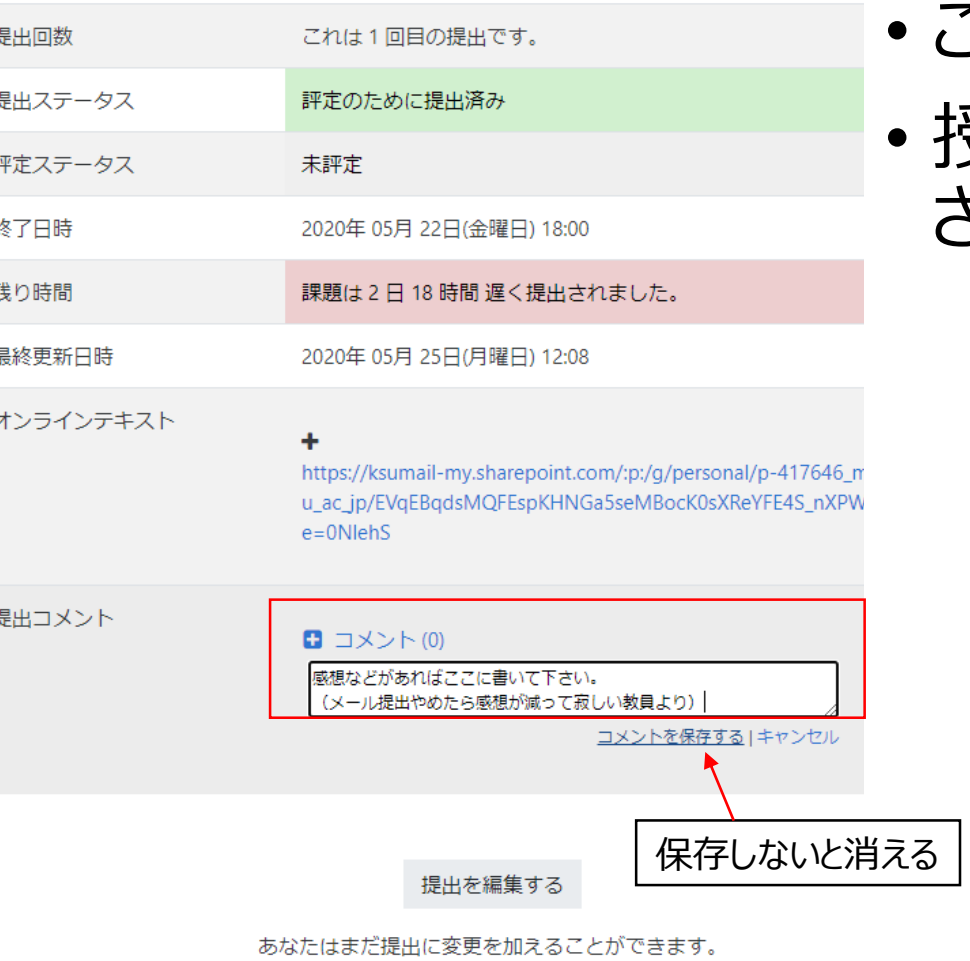
採点されると、「評定済み」になる

採点＝教員（私）が手動で点数を付ける

さらに下に、評点とフィードバックコメント（教員からのコメント）

毎回、何点だったかを確認しておこう（評点アップの交渉も遠慮せずにどうぞ）
（1回と2回はそのうち追加します(余裕が出来れば)）

Moodleのコメントについて



- こんな感じでコメント書けます。
- 授業の感想などはここに書いて下さい。
  - ただし、コメントへの返信は期待しないで下さい。
    - 返信欲しいことはメールで聞きましょう。
- 質問もメール等で！
  - レポートの採点はすぐに出来る訳ではありません（それなりに大変で時間もかかります）

そろそろ折り返しなので・・・

- 単位の取得とレポートについて
  - 難しい、で諦めないようにしましょう
    - 前も書いたけど、
      全員がしっかり分かってるわけではないです
    - 細かいこと考えるより前に、プログラムを書きましょう
      - そのうち分かってくればよい
  - 少し変えただけ、でも最低基準は満たす
    - ただし、小テストはしっかりやっておきましょう
  - 最後の課題をしっかり作ればいい成績も取れる
    - Sを取れるかはさすがに別になりますが

レポート（PowerPoint）の注意

- スライドにアニメーションを付けないこと：採点が大変になるので
  - 最終回に提出するレポートは別
- 「プログラム」スライドの目的
  1. 採点のため：動かないプログラムは評価出来ない
  2. バックアップのため
     - どこに保存したか分からない　→　Moodleにあります
  3. プログラムの再利用
     - 前回までのプログラムをコピペで応用
- コピー＆ペーストでProcessingで動かせることが大事
  - Moodleからレポートをダウンロードし、プログラムをコピー＆ペーストで動かせればOK
  - Web版のOffice365でスライドを開くと、
    コピペで動かない場合があるので、一度ダウンロードして、
    アプリ版のPower Pointで開いてコピペすること

プログラムスライドについて

- 動かないプログラムを送ってきても評価できません

- 動く状態になってから、プログラムをコピー、貼り付けましょう。

- **「全て選択」「Select All」** → コピー
  - マウスやタップで手動で選択しないように！
    - 何回も貼り付けをすると、文字が消える
  - PowerPointでプログラムを修正しないように！
    - 貼り付けた後は余計なことはしない

今回の授業内容

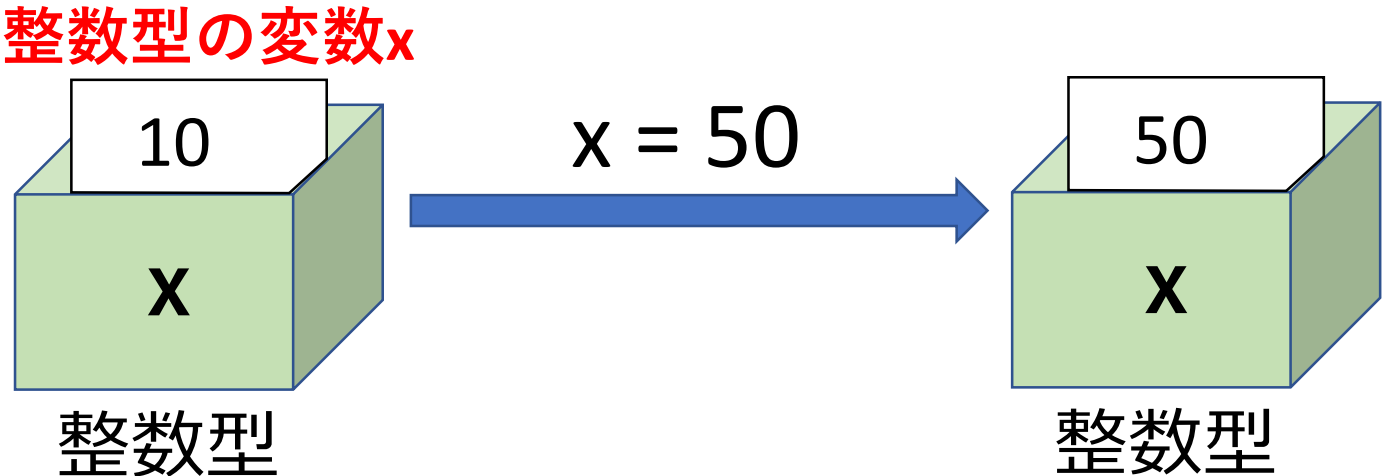
- 条件分岐（if文）
  - 条件によって動きを変えるプログラムを作る
    - 時間変化
    - 座標変化
- レポート内容
  - 条件分岐を使ったプログラムを提出

- 少し注意：条件分岐は少し難しい
  - とにかく慣れるまでは、
    よく分からなくても動かして試すを繰り返しましょう
  - 深く理解できなくても、今は少し変えて動きが変わったでOK

復習

変数：値を保存できる箱のようなもの

- 変数:数値や文字列を格納（保存）しておける
- 変数には型と名前がある
  - 変数の型　：　格納できるデータの種類
  - 変数の名前　：　変数を区別するための名前
- 変数の中身は変えることが出来る



変数の型と名前

- 変数の型：数値や文字など色々な型がある
  - 整数：int
  - 小数：float
  - 文字列：string
- 変数の名前：自分で好きに付けられる
  - 付けられない変数名
    - 数字から始まる名前
    - _(ｱﾝﾀﾞｰｽｺｱ)以外の記号を含む名前
- 変数名も大文字、小文字はきっちり区別
  - **int X; と int x; は全く別の変数**になる

変数の宣言

- 変数を作ることを、変数の宣言という
- 宣言の仕方

変数の型　変数の名前;

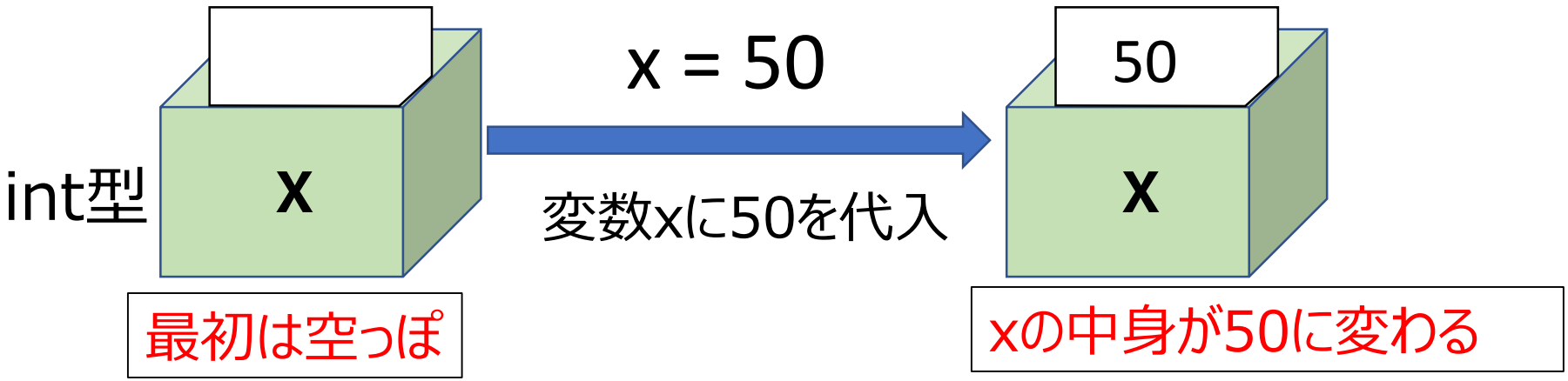
- 整数型の変数 x を宣言
  int x;
- 小数型の変数 yとz を宣言
  float y, z;

型が同じなら、複数まとめて宣言できる

代入

- 変数に値を入れることを代入という
- 代入の仕方

変数の名前　＝ 値;



※正確には、空っぽではなく初期値がはいる (int の場合は0)

変数に式を代入

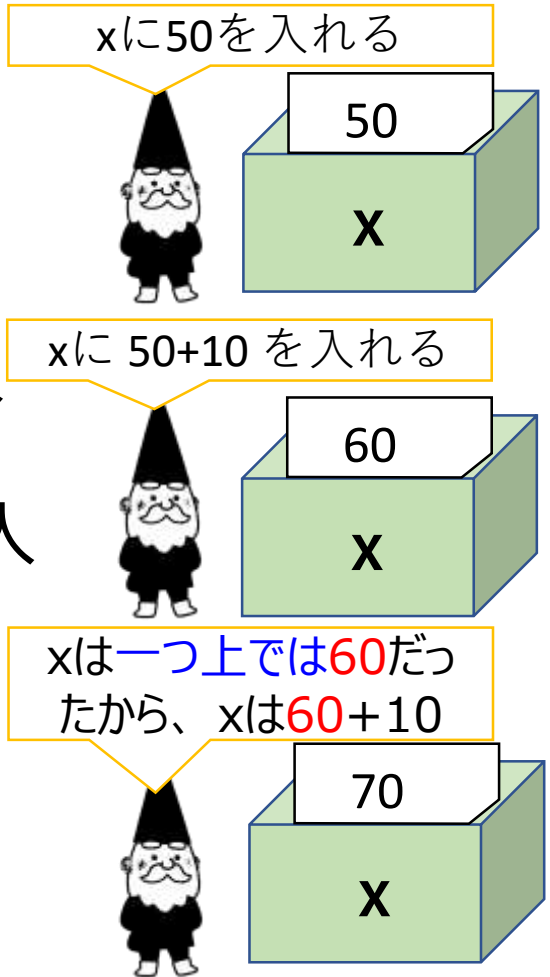
- 数値型の変数には、数だけでなく式も代入できる

```
int x;      //整数型の変数xを宣言
x = 50;  //xに50を代入

x = 50 + 10; //xに(50+10)を代入

x = x + 10; //xに(元のx+10)を代入
```

代入する式にその変数を使うと、
変数の数を増やしたり減らしたり出来る



演算子

- 基本的な演算子

演算名	演算子	例
足し算	+	x = 10 + 1;
引き算	-	x = 10 - 3;
掛け算	*	x = 5 * 4;
割り算	/	x = 10 / 2;
割った余り	%	x = 2 % 1;

- 資料では、この演算子のみで説明する
- 次ページの演算子は、使うと少しだけ楽になる

便利な演算子

- 覚えておくと便利な演算子

演算名	演算子	例	別の書き方
加える	+=	x += 5;	x = x + 5;
引く	-=	x -= 2;	x = x – 2;
かける	*=	x *= 3;	x = x * 3;
割る	/=	x /= 10;	x = x / 10;
インクリメント（1増やす）	++	x++;	x = x + 1;
デクリメント（1減らす）	--	x--;	x = x – 1;
負数	-	x = -x;	x = x * -1;

楽になるだけで、使えないと困るわけではない

変数を使った図形の移動や変形

- 例えば、右に動く丸を描くプログラム

```
int x = 50;  //図形のx座標を管理する変数
void setup(){
  size(400, 400);
}
void draw(){
  background(255);
  ellipse(x, 200, 50, 50);
  x = x + 2;  //xの値を2ずつ増やす（右に2ずつ移動）
}
```

- 右端まで移動したら消えてしまう

条件によって変わる動きを実現する

- 座標によって動きを変える
  - もしも、画面外に出たら、最初の場所に戻る
    - 動きをループさせる
  - もしも、画面外に出たら、逆方向に動くようにする
    - 条件に応じて移動する方向を変える
- 経過時間によって動きを変える
  - もしも、5秒たったら、色を変える
  - 10秒ごとに図形の形を変える

- 他にも色々出来るが、今回は主にこの2点

if文：条件分岐

- 構文

```
if(条件) {
    条件を満たした場合の処理
}
```

比較演算子

数学の記号と同じではないことに注意

演算子	==	!=	>	>=	<	<=
意味	等しい	異なる	大きい	以上	小さい	以下

比較演算子を使った式を、論理式という

論理式とboolean型

- 論理式の結果は、正しい(真)か、正しくない(偽)か
  - 論理式が正しい（条件を満たしている）
    ⇒　結果は true
  - 論理式が正しくない（条件を満たしていない）
    ⇒　結果は false
- trueかfalseか、2つに1つ

- boolean型：true,falseを代入できる変数型

if-else if- else文:複数の条件分岐

- 構文

```
if(条件１) {
   条件１を満たした場合の処理;
} else if(条件２){
   条件１を満たさずに、
   条件２を満たした場合の処理;
} else {
   どの条件も満たさない場合の処理
}
```

条件分岐文の例

- もしも、x が 200 より大きければ、xを0にする

```
if(x > 200) {
  x = 0;
}
```

- もしも、count が 100以上なら、背景を黒くする

```
if(count >= 100) {
  background(0,0,0);
}
```

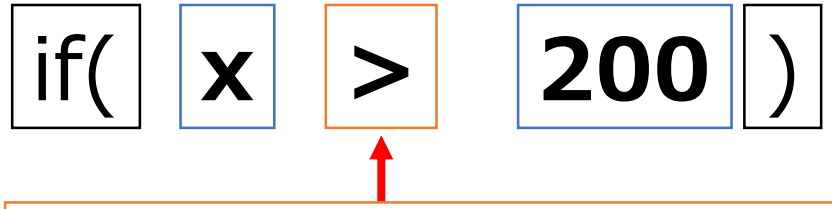
- もしも、hue が 360以上なら、hueを0にする

```
if(hue >= 360) {
  hue = 0;
}
```

比較演算子：どっちが大きいか？

- もしも、x が 200 より大きければ、xを0にする

```
if(x > 200) {
  x = 0;
}
```



> の広い側が x に向いている ⇒ x の方が大きい

書いてみて、＜ や ＞ の広い方が、どっちを向いているかを確認
逆だったら、逆の記号に書き換えればよい

if文を書く時の注意

- **() {} は必ず開いたら閉じる**
- **閉じすぎにも注意**
- **カッコの対応が付いているか、しっかり確認！！**

条件によって変数の値を変える

- もしも、 x が width より大きければ、xを0にする

```
if(x > width) {
  x = 0;
}
```

  - もしも、xが画面外に出たら、最初の位置に戻す

- もしも、 y が 0 より小さければ、yをheightにする

```
if(y < 0) {
  y = height;
}
```

  - もしも、yが画面外に出たら、最初の位置に戻す

if文を使ったループ

```
int x = 50;  //図形のx座標を管理する変数
void setup(){
  size(400, 400);
}
void draw(){
  background(255);
  ellipse(x, 200, 50, 50);
  x = x + 2;  //xの値を2ずつ増やす（右に2ずつ移動）

  if(x >= width){  //もしも、xがwidth(画面幅)以上なら = 画面外なら
    x = 0;  //xに0を代入
  }

}
```

if文を使ったループ(画面端以外)

```
int y1 = 350;  //図形のx座標を管理する変数
void setup(){
  size(400, 400);
}
void draw(){
  background(255);
  ellipse(200, y1, 50, 50);
  y1 = y1 - 2;  //y1の値を2ずつ減らす（上に2ずつ移動）

  if(y1 <= 100){  //もしも、y1が100以下なら
    y1 = 350;  //y1に350を代入
  }

}
```

時間経過で動きや形を変える

- 準備：時間をカウントする変数を宣言してインクリメント

```
int count = 0; //時間を管理する変数
void setup(){
  size(400, 400);
  frameRate(30);  //1秒で30回更新するように設定
}
void draw(){
  count++;  //drawで毎回1ずつ増やす
  background(255);
  fill(255,0,0);
  ellipse(200, 200, 50, 50);
}
```

- count++; は count = count +1; と同じ意味

3秒たったら色が変わる

```
int count = 0; //時間を管理する変数
void setup(){
  size(400, 400);
  frameRate(30);  //1秒で30回更新するように設定
}
void draw(){
  count++;  //drawで毎回1ずつ増やす
  background(255);
  if(count < 90) {  //3秒(3*30)未満なら、
    fill(255,0,0);  //赤色で塗りつぶし
  } else {  //そうでなければ ( 3秒 以上なら)
    fill(0,0,255);  //青色に塗りつぶす
  }
  ellipse(200, 200, 50, 50);
}
```

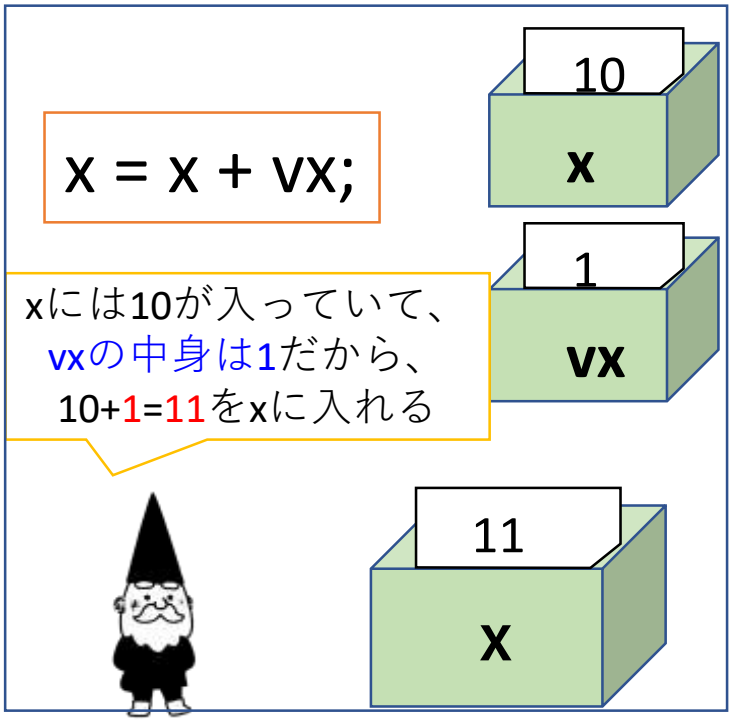
変数に変数を足す

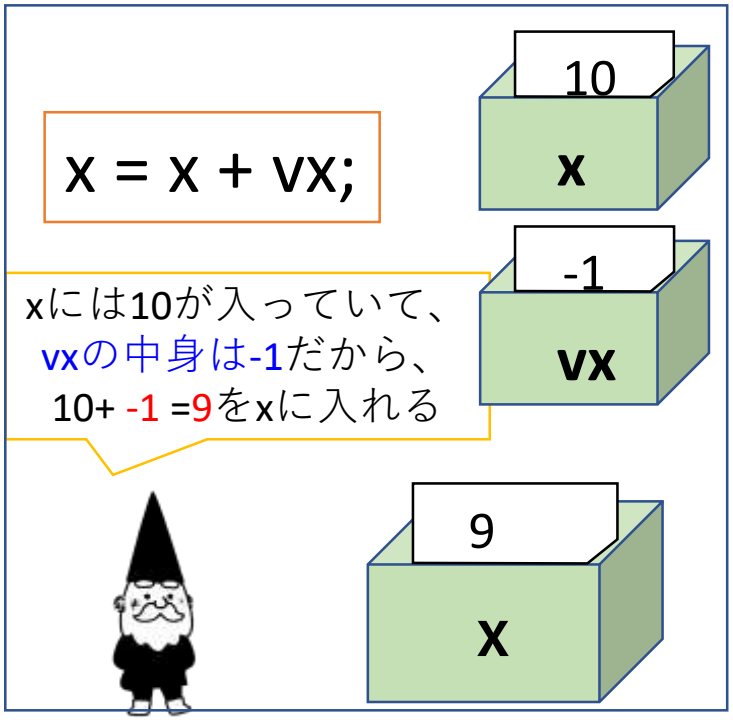
- ある程度進んだら加速する図形

```
int x = 0;  //図形のx座標を管理する変数
int vx = 1;  //図形のx座標方向の移動速度を管理する変数
void setup(){
  size(400, 400);
}
void draw(){
  background(255);
  ellipse(x, 200, 50, 50);
  x = x + vx;  //xにvxを足す
  if(x >= width/2){ //もしも、xが画面半分(width割る2)以上なら
    vx = 3;  //vxに3を代入
  }
}
```

変数に変数を足す

- 変数に足す変数の値を変えると、同じ式でも結果が変わる





同じ式でも、足す変数の値が変わると結果が変わる

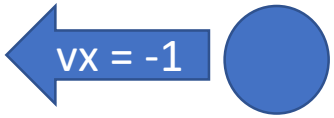
図形を跳ね返すには？

- x に vxを足す（xは毎フレーム vx だけ増減)
  - vxがプラスなら右へ、vxがマイナスなら左へ

- 最初左から右へ移動する（vx ＝ 1)
- 画面外に出たら左に移動するようにする
  - 画面外：x がwidth以上
  - もしも、xの値がwidth以上なら vx を -1にする
    - vxが-1になると、xは1ずつ減る ⇒ 左に動く

- 最初は右に進む

- もしも右端まで来たら、vxの符号を逆転させる



- vx が-1になると、今度は左に進む

- もしも左端まで来たら、vxの符号を逆転させる

- vx が1になると、右に進む

図形を跳ね返るようにする

- 画面外に出たら移動速度(vx)を逆転させる

```
int x = 0;
int vx = 1;
void setup(){
  size(400, 400);
}
void draw(){
  background(255);
  ellipse(x, 200, 50, 50);
  x = x + vx;  //xにvx(移動速度)を足す
  if(x > width){  //もしも、xが右端まで行ったら
    vx = -1;  //xの移動速度を-1(左方向)に変える
  } else if(x < 0) {
    vx = 1;  //xの移動速度を1(右方向)に変える
  }
}
```

ある範囲内で跳ね返るようにする

- 画面外に出たら跳ね返る

```
if(x < 0) {
  vx = 1;
} else if(x > width) {
  vx = -1;
}
```

- 50～100の範囲を出たら跳ね返る

```
if(x < 50) {
  vx = 1;
} else if(x > 100) {
  vx = -1;
}
```

※xの初期値を 50~100にすること

y方向についても、同様にすれば、画面内を斜めに跳ね返るようになる

アニメーションの途中で一時停止

- 下記のプログラムを、一番下に追加
  - draw(){}やsetup(){}の中に入れないように
  - 必要なら入れる程度で、入れなくても良い

```
boolean stop = false;
void mousePressed(){
  if(stop==false){
    noLoop();
    stop = true;
  } else {
    loop();
    stop = false;
  }
}
```

今回のレポート

- 条件分岐を使ったプログラムを提出
  - 最低基準
    - 色付きの動く図形が3つ以上ある
      - それぞれ違う色、違う形にすること
      - それぞれ違う動きにすること
    - if文を最低3つ使っている
  - Power Pointでスライドを作って提出

- 加点項目
  - 独自の動きを追加した
  - 過去回のプログラムに条件分岐を取り入れた

アレンジのヒント

- 線ではなく他の図形も描いてみる
  - line() を ellipse()や rect()に変える
- 回転角度やノイズのシード値の増加量を変えてみる
- 中心座標もランダム(ノイズ)にしてみる
  - line(0,0,x,y); の0,0もノイズで動かす
    - 変数を一通り追加する
- 座標をマウス座標にしてみる
  - line(mouseX,mouseY,x,y);
- rect(x,y,s3,s3); のように、自分で宣言した変数を使う

PowerPointでレポート作成

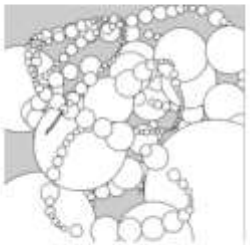
- 1枚目：タイトル

PC基礎 05

学籍番号　氏名

- 2枚目：実行画像

タイトル：（作品のタイトル）



- 説明
  - ～～をイメージして作った
  - マウスの動きに合わせて、丸の大きさが変わるように工夫した

どんなものを作ろうとしたのか？

どんな工夫をしたのか？

簡単でいいので説明を書く

- 3枚目：プログラム

プログラム

プログラムを全て貼り付ける

```
void setup(){
  size(400,400);
  colorMode(HSB,360,100,100,100);
}
void draw(){
  float d = dist(mouseX, mouseY, pmouseX, pmouseY);
  ellipse(mouseX,mouseY,d,d);
}
```

貼り付けたときに、文字が見えない程小さくなっても構わない

これは第3回のレポートの例

実行画面やプログラムは、
（当然）今回の内容にすること

必要事項が揃っていれば、レイアウトは自由

レポートチェックリスト（第7回）

- ミニテストを受験した(レポート提出の前でも後でも可)

- スライドのサンプルプログラムをアレンジした
  - 動く図形が3つ以上ある
  - if文が3つ以上ある
- PowerPointでレポートを作成した
  - タイトル、作品紹介(工夫点)、プログラムの3枚
- Moodleでレポートを提出した