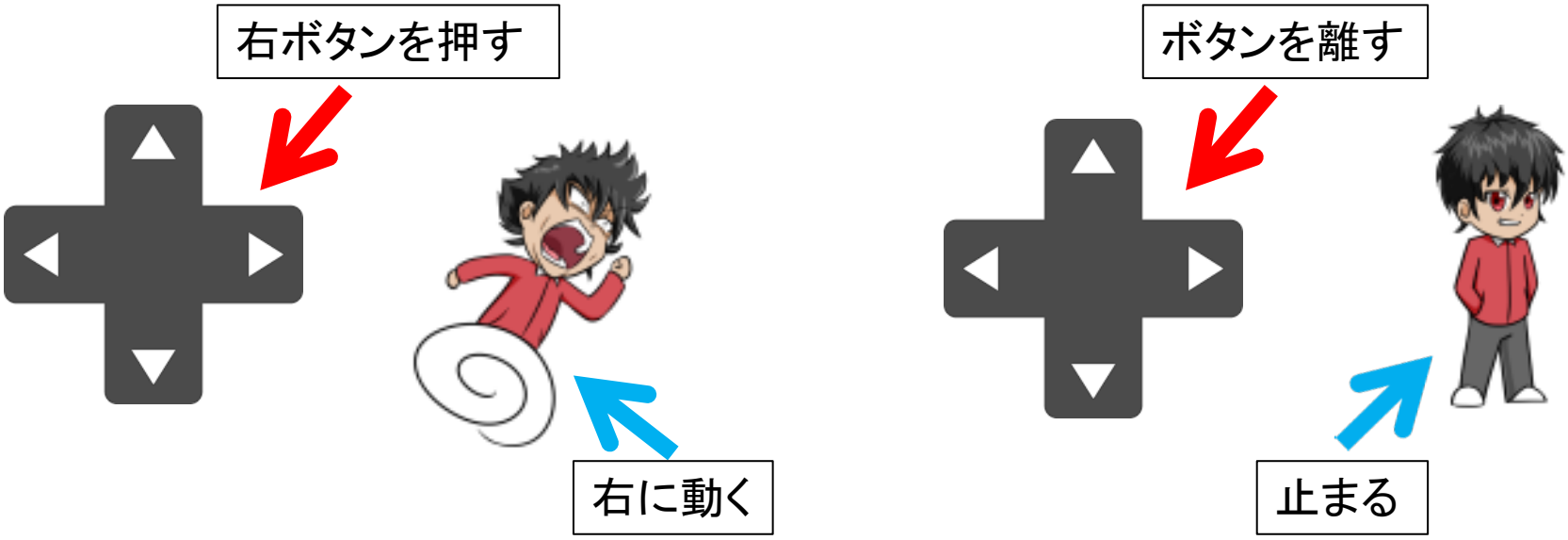
プログラマー勉強会第2回

条件を満たすときだけ実行する

今までのプログラムは記述したものは全て実行されてきたが、
特定の条件を満たすときだけ、処理を実行するプログラムも必要になってくる。

[例]



右ボタンを押された時に、プレイヤーを動かす処理を実行し、
押されていないときはプレイヤーを止める処理を実行する。

値に応じて出力が変わる

emacs cond.c&　を入力

[cond.c]

```
#include <stdio.h>

int main(){
    int n;
    printf(“値を入れてください：”);
    scanf(“%d”,&n);
    if(n)
        printf(“ifの中を通りました\n”);
    else
        printf(“elseの中を通りました\n”);
    return 0;
}
```

0以外を入力　→　ifの中を通りました
0を入力　→　elseの中を通りました

if文else文と条件式

if文・・・条件式が正しい時、プログラムを実行させる。
else文・・・if文の下に置く。if文の条件式が正しくない時に
　　プログラムを実行させる。
条件式・・・if文などに置く、プログラムの実行条件を示す式
　　0以外の数値の時正しい(真)、0の時正しくない(偽)を
　　意味する。

[使い方]

```
if(条件式)
    プログラム;   ←  条件式が真のとき実行するプログラム
else
    プログラム;   ←  条件式が偽のとき実行するプログラム
```

[例]

```
if(n)
    printf(“ifの中を通りました\n”);
else
    printf(“elseの中を通りました\n”);
```

if文の中が複数行になる時

if文の条件式を満たした時に実行する文が2行以上になる場合はその部分を{}で囲む。elseの場合も同様。

[例]

```
if(n){
    printf(“ifの中を通りました\n”);
    m = n + 3;
    printf(“%d\n”,m);
}
```

条件演算子

条件演算子・・・条件式に使用する演算子。
条件を満たすときに1、満たさない時に0を出す。

[等値演算子]

== ・・・左の値と右の値が等しいかどうか(=)

→ if(n == 3)・・・nが3かどうか？

!= ・・・左の値と右の値が等しくないかかどうか(≠)

→ if(m != -2)・・・mが-2でないかどうか？

[関係演算子]

< ・・・左の値が右の値より小さいかどうか(<)

→ if(n < 8)・・・nが8より小さいかどうか？

> ・・・左の値が右の値より大きいかどうか(>)

→ if(n > -4)・・・nが-4より大きいかどうか

条件演算子②

<= ・・・左の値が右以下かどうか(≦)

→ if(n <= m)・・・nがm以下かどうか？

>= ・・・左の値が右以上かどうか(≧)

→ if(n >= -m)・・・nが-m以上かどうか？

[論理演算子]

&& ・・・左の条件式と右の条件式が共に真かどうか(∩)

→ if(1 <= m && m <= 3)・・・mが1以上かつ3以下かどうか？

|| ・・・左の条件式と右の条件式がどちらか真かどうか(∪)

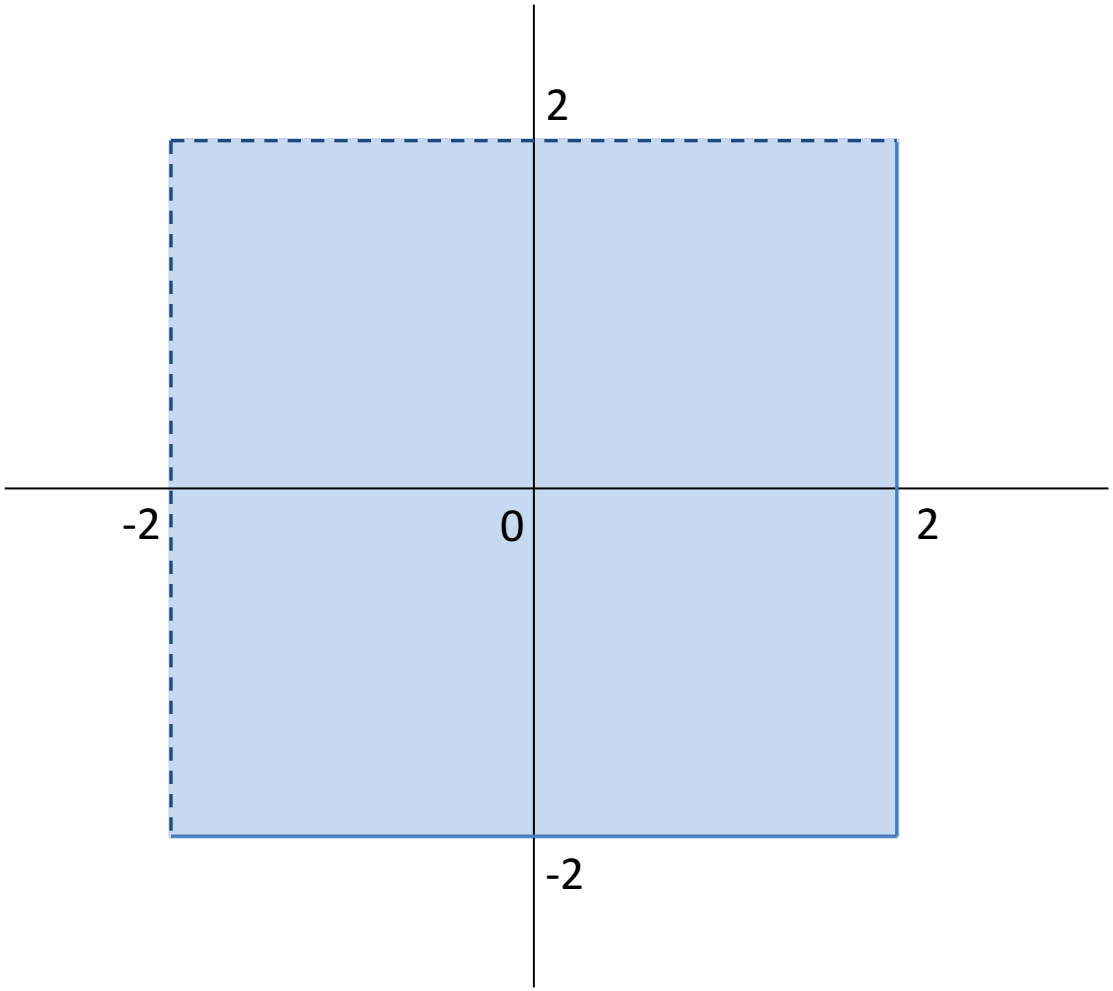
→ if(m > 3 || m < -1)・・・mが3より大きいもしくは-1より小さいかどうか？

! ・・・条件式を反転させる

→ if(!(m == 3))・・・mが3かどうか？の否定 → m != 3と同じ意味

演習（領域内にあるか）

入力された2点が青い領域内にあるかどうか判定するプログラムを作ってください
点線は領域に含まず、実線は含むものとする。(ファイル名　ex2-1.c)



[実装の手順]
scanfで2点を入力
↓
その2点と領域の端の値を
if文で比較。
↓
条件式には比較演算子と
論理演算子を使う

3つ以上の比較

if~else文では選択肢が2つしかなかったが、選択肢が3つ以上必要な場合もある。

if~elseのelseの次にif をつなげる。

[例]

```
if(n < 3)
    printf(“n < 3”);
else if(3 <= n && n <= 14)
    printf(“3 <= n <= 14”);
else
    printf(“n > 14”);
```

switch文

3つ以上の場合分けで、番号(整数)による場合分けを行う時はif~elseでも可能だが、switch文というものがある。

[使い方]

```
switch(int型の変数){
case 数値:
    プログラム;
    break;
case 数値:
    プログラム;
    break;
…
default:
    プログラム;
    break;
}
```

変数内の数値と同じ数値のcaseへ飛ぶ
該当する数値がない場合はdefaultへ飛ぶ
case内の最後にはbreak文を置く。
置かないとそのまま下のcaseが処理されてしまう。

break・・・switch文などを途中で抜け出す処理。

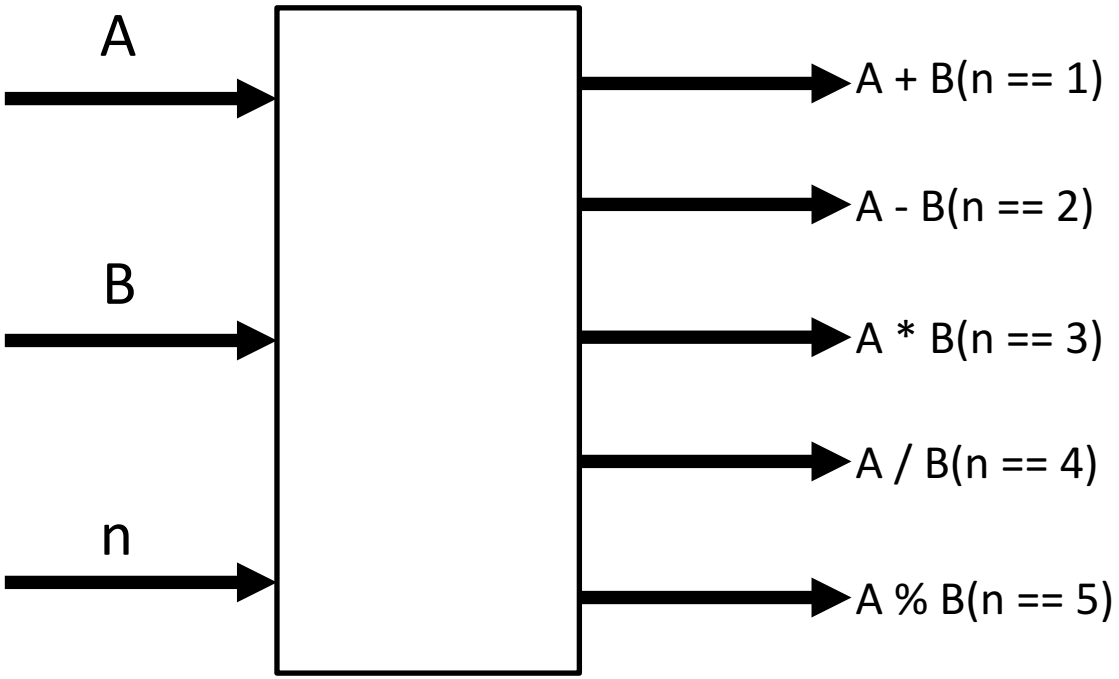
switch文②

[例]

```
switch(n){
case 1:
    printf(“case = 1\n”);
    break;
case 2:
    printf(“case = 2\n”);
    break;
case 3:
    printf(“case = 3\n”);
    break;
default:
    printf(”case is not 1 ～ 3\n”);
    break;
}
```

演習（ALUみたいなやつ）

変数A、Bを入力した変数nによって演算方法を変えるシステムを実装する。
（ファイル名　ex2-2.c）



[実装の手順]
scanfでA,Bの値を入力
Bは0以外の数値を入れる
↓
scanfでnの値を入力
↓
switch文もしくはif~elseを用いて、nの値に応じて演算方法を変え、その結果を出力。
↓
nが該当する値でない場合、「該当する演算がありません」と出力する。

ループ

プログラム中では同じ処理を何度も実行する必要がある。
条件式が真である限り、処理を繰り返し実行し続けるwhile文と言うものがある。

[使い方]

while(条件式)
　　繰り返し実行するプログラム;

[例]

変数nが100以下である限り、nに3を加え、nの値を出力し
続けるプログラム。

```
while(n <= 100){
    n += 3;
    printf(“n = %d”,n);
}
```

無限ループとループの脱出

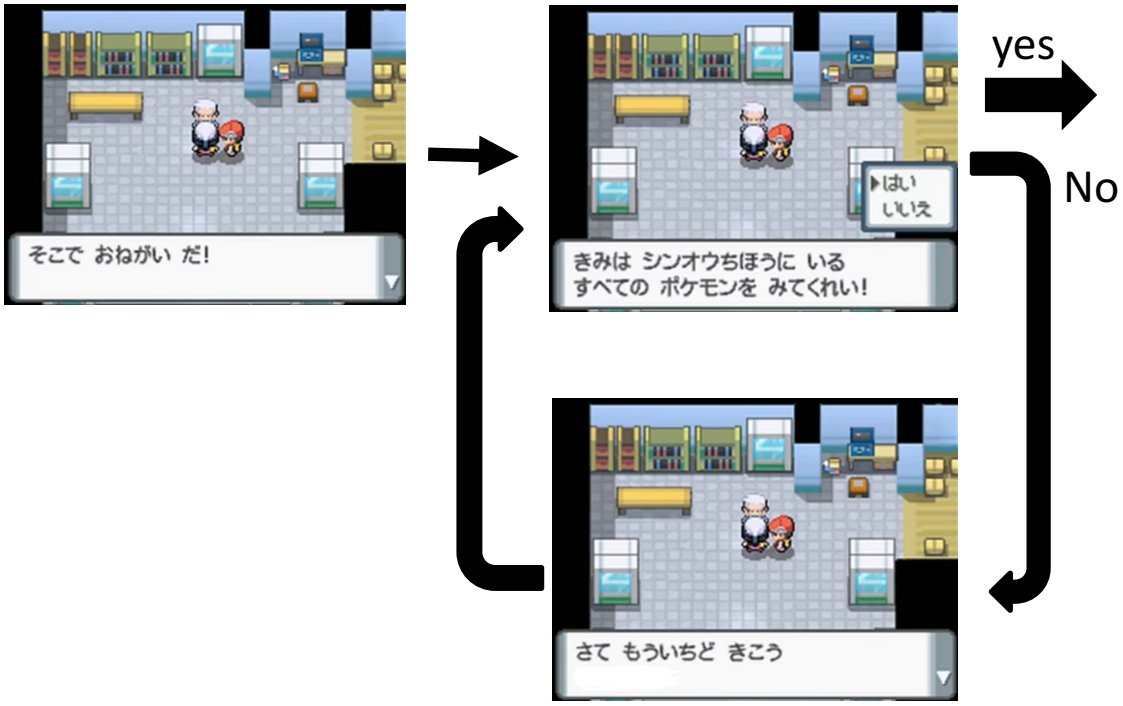
[無限ループ]
whileの条件式の中身を1とする　→　while(1)
条件式が常に真となり、無限ループとなる。

[例]
```
while(1)
    printf(“竹書房は指定暴力団\n”);　→　永遠にこの文字列が表示される。
```

[ループの途中脱出]
条件式は満たしているが、直ちにループを脱け出したい時
→　switch文で使ったbreak文を使う。
　　そのまま置いては意味がないので、if文などと組み合わせて使う。
[例]
```
while(1){
    n += 3;
    if(n == 10)
        break;　→　nが10になった時ループを脱け出す
    n -= 2;
}
```

演習（断らせない）

ポ〇ケットモンスターというゲームで、図鑑を博士から渡される時拒否すると「そんなこと言わずに」と言われて承諾するまで永遠に問われ続け、ゲームが進行しなくなる~~バグ~~システムがある。while文、if文、scanfでこのシステムを実装できるので、作成してください。(ファイル名　ex2-3.c)



[実装の手順]

whileループへ入る

↓

質問を問う

↓

「はい」を1、「いいえ」を0などと置いてscanfで入力させる

↓

「はい」だった場合ループを脱け出す

↓

「いいえ」だった場合脱け出さない

回数指定のループ

while文が条件を満たす限り何度でも処理を実行する文だったのに対し、あらかじめ回数が分かっていて、その回数分だけ処理を実行させる文としてfor文がある。

[使い方]

for(カウンタ変数の初期値の設定;条件式;カウンタ変数の処理)
　　繰り返し実行するプログラム;

[例]

Hello Worldを10回出力するプログラム

```
for(i = 0;i < 10;i++)
    printf(“Hello World\n”);
```

カウンタ変数・・・for文内でループの回数を保存する変数

カウンタ変数の処理にはインクリメントやデクリメントだけでなく、i += 2
i *= 3のように使うこともできる。

演習（倍数の表示）

for文を使って、1～100の間で指定された数の倍数のみを出力するプログラムを作成してください。(ファイル名　ex2-4.c)

2のとき
→2,4,6,…100

3のとき
→3,6,9,…99

10のとき
→10,20,30,…100

[実装の手順]
scanfで指定する数を入力
↓
forループへ入る
↓
カウンタ変数にインクリメントを効かせ、if文などを使って表示
or
カウンタ変数に対する処理を変えてみる

たくさんの変数が必要

今まで、変数は2、3個ほどしか登場しなかったが、実際のプログラムでは数十から数百、場合によっては数万ほどの変数(データ)が必要になる時がある。
そんな時、大量の変数を宣言するのは不可能である。

[10個宣言]

[1000個宣言]

[1145141919810931個宣言]

→　一度に大量の変数を宣言するために配列というものを使う

配列

配列・・・同じ型の変数を多数宣言して、番号によって管理する仕組み

[使い方]
変数の型名 配列名[要素数];
　→　配列名のルールは変数名と同じ
　要素数・・・配列として宣言したい変数の数

[例]
int型の要素数10の配列を宣言
int array[10];　→　int型の変数10個が宣言されたのと同じ意味。

[初期化]
変数の型名 配列名[要素数] = {要素1,要素2,要素3,…};
　→　配列の該当する場所に数値が格納される
　→　int array[5] = {1,5,6,3,2};
変数の型名 配列名[要素数] = {要素};
　→　全ての要素を同じ数値で初期化する。
　→　double array[10] = {0.5};

配列②

[参照]

配列名[参照したい要素番号]

→ int array[10];と宣言された配列の２番を参照したい。

→ array[2];

→ int型の変数でも参照可

→ n = 3;
　　array[n]; → ３番が参照される。

要素番号・・・配列の要素を参照するときの０～要素数−１までの番号。

→ 要素番号は0から始まるので、要素番号の最大値は要素数−１となる。

→ int array[7]; → array[7]というものは存在しない。
　　array[0],array[1],array[2],…array[6]までの7個

配列とfor文

配列の要素番号はint型の変数でも参照できるので、カウンタ変数があるfor文と相性がいい。

[例]
配列の全要素を表示

```
int array[10] = {0,5,-2,7,9,12,2,-8,2,-6};
int i;

for(i = 0;i < 10;i++)
    printf(“%d番：%d\n”,i,array[i]);
```

→ for文とprintf一行だけで、10個の変数の値が表示できる。

演習（合計値、内積）

(1)int型の要素12の配列Aがある。
A[12] = {0,2,8,-1,8,9,12,-4,3,6,7,-14};
この配列全要素の合計値を求めるプログラムを作ってください。
（ファイル名　ex2-5-1.c）

(2)double型の要素3の配列A,Bがある。
A[3] = {2,0.7,-1.2};
B[3] = {-0.5,1.7,4};
この二つの配列をベクトルとして考え、二つのベクトルの内積を
求めるプログラムを作ってください。
（ファイル名　ex2-5-2.c）

演習の答え

[ex2-1.c]

```
#include  <stdio.h>

int  main(){
int  X,Y;

printf("２点を入力してください：");
scanf("%d%d",&X,&Y);
if(-2 < X && X <= 2 && -2 <= Y  &&  Y  <  2)
    printf("(%d,%d)は領域内にあります\n",X,Y);
else
    printf("(%d,%d)は領域内にありません\n",X,Y);
return  0;
}
```

演習の答え

[ex2-2.c]

```
#include  <stdio.h>
int  main(){
    int  n;
    int  A,B;

    printf("A,Bの値を入れてください：");
    scanf("%d%d",&A,&B);
    printf("nの値を入れてください(1~5)：");
    scanf("%d",&n);

    switch(n){
    case  1:
        printf("%d\n",A  +  B);
        break;
    case  2:
        printf("%d\n",A  -  B);
        break;
```

演習の答え

```
    case 3:
        printf("%d\n",A * B);
        break;
    case 4:
        printf("%d\n",A / B);
        break;
    case 5:
        printf("%d\n",A % B);
        break;
    default:
        printf("該当する演算はありません\n");
        break;
    }
    return 0;
}
```

演習の答え

[ex2-3.c]

```
#include  <stdio.h>
int  main(){
    int  n;
    printf("お願いだ\n");
    while(1){
        printf("図鑑を受け取ってくれ\n");
        printf("はい：1　いいえ：0 →　 ");
        scanf("%d",&n);
        if(n  ==  1)
            break;
        else
            printf("まあそう言わずに\n");
    }
    printf("ありがとう\n");
    return  0;
}
```

演習の答え

[ex2-4.c]

```
#include  <stdio.h>
int  main(){
    int  i,n;
    printf("数を入れてください：");
    scanf("%d",&n);
    for(i  =  n;i <= 100;i += n)
        printf("%d  ",i);
    printf("\n");
    return  0;
}
```

```
#include <stdio.h>
int main(){
    int i,n;
    printf(“数を入れてください：”);
    scanf("%d",&n);
    for(i = 1;i <= 100;i++){
        if(i % n == 0)
            printf("%d ",i);
    }
    printf("\n");
    return 0;
}
```

演習の答え

[ex2-5-1.c]

```
#include  <stdio.h>
int  main(){
    int  A[12]  =  {0,2,8,-1,8,9,12,-4,3,6,7,-14};
    int  i;
    int  sum  =  0;

    for(i  = 0;i < 12;i++)
        sum  +=  A[i];

    printf("Aの合計は%d\n",sum);
    return  0;
}
```

演習の答え

[ex2-5-2.c]

```
#include  <stdio.h>
int  main(){
    double  A[3]  =  {2,0.7,-1.2};
    double  B[3]  =  {-0.5,1.7,4};
    double  in  =  0;
    int  i;

    for  (i = 0;i < 3;i++)
        in  +=  A[i]  *  B[i];

    printf("%f\n",in);
    return  0;
}
```