プログラミングの基礎及び演習

情報工学科

シラバス

3,4. 分岐処理の実現方法及び復習
関係演算子を用いた分岐処理の実現方法の確認
事前学修：授業スライド「第2回　分岐処理の実現方法及び復習」と教科書p.45～p.55の熟読。（60分）
事後学修：授業スライドの用語とその意味の理解。分岐処理のフロー，if-else文やswitch文によるプログラミング法，論理演算子を用いた条件の書き方を説明できること。（60分）
演習課題のプログラム及びレポートの完成と提出。（180分）

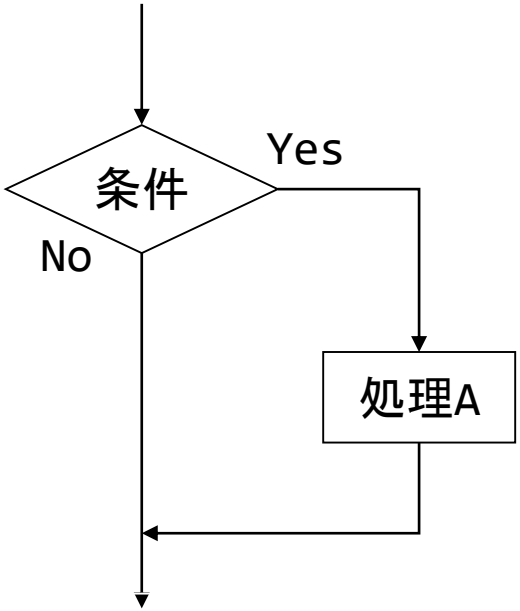
プログラミングの基礎　第2回

- 分岐処理
  - 分岐処理の復習
  - 分岐処理をC言語で表現するには？
    - if文
    - 関係演算子
    - 複雑なif文
    - 複雑な条件式
    - switch文
  - 教科書p.45～p.60参照

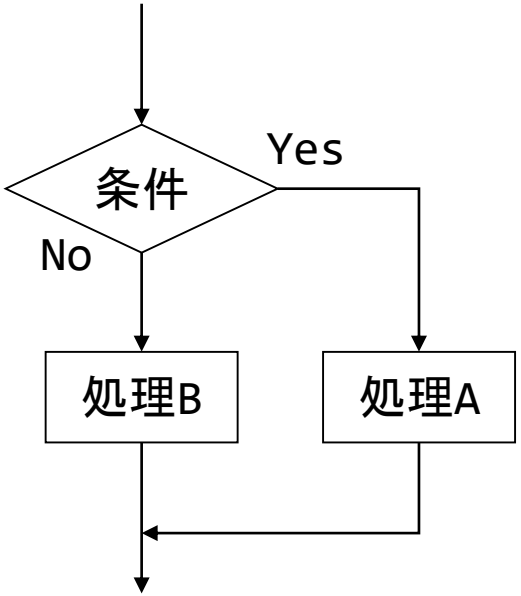
分岐処理とは？

- ある条件を調べ，その結果に応じて手順が変化する
  - 例：
    - 喉が渇いていたら水を飲む．乾いていなければ飲まない
    - 2つの変数xとyがある．
      x >y ならば"xの値が大きい"と表示する．
      それ以外なら"yの値がxの値以上である"と表示する
  - このような例は実際のプログラムの中で多数出現する

分岐処理のフローチャート



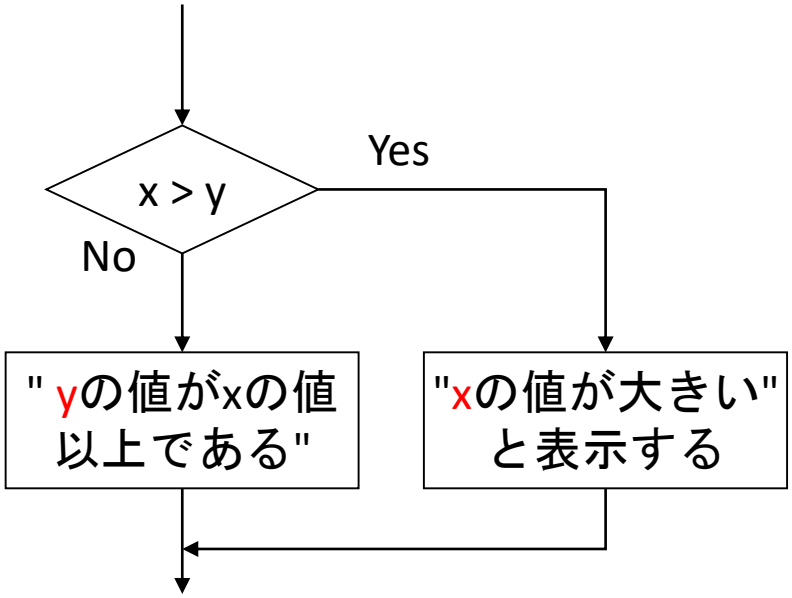
「条件」が真なら，
「処理A」を実行
それ以外なら，何もしない．



「条件」が真なら，
「処理A」を実行
それ以外なら，
「処理B」を実行

フローチャートの例

- 2つの変数xとyがある
  - x>yならば"xの値が大きい"と表示
  - それ以外なら" yの値がxの値以上である"と表示

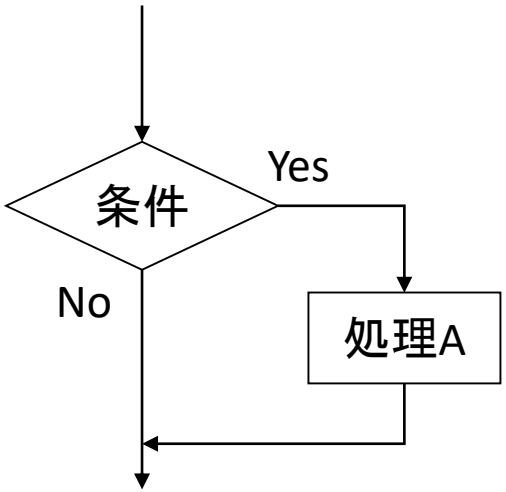


プログラミングの基礎　第2回

- 分岐処理
  - 分岐処理の復習
  - 分岐処理をC言語で表現するには？
    - if文
    - 関係演算子
    - 複雑なif文
    - 複雑な条件式
    - switch文
  - 教科書p.45～p.60参照

C言語での分岐処理

- C言語では，分岐処理をif文で表現する
  - 書式：
    - if (条件)
      処理A;

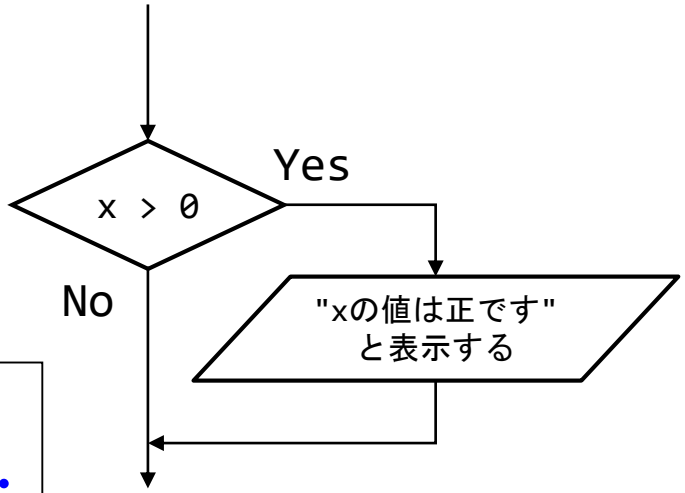


「条件」は括弧（）で囲む
「処理A」の後ろにはセミコロンが必要
（条件）の後ろにはセミコロンはつけない

if文の例 (1)

- 変数xの値が正であれば，“xの値は正です”と表示する

```
if ( 条件 )
    条件が真の時の処理;
```




```
if ( x > 0 )
    printf("xの値は正です¥n");
```

プログラムを設計した時に分岐処理が使われていれば，
それをif文に置き換えれば良い

条件の書き方 （関係演算子）

a < b	aがbより小さいなら真， それ以外なら偽	
a > b	aがbより大きいなら真， それ以外なら偽	
a <= b	aがb以下なら真， それ以外なら偽	a ≦ b という意味. a =< b ではダメ.
a >= b	aがb以上なら真， それ以外なら偽	a ≧ b という意味 a => b ではダメ.
a == b	aとbが等しいなら真， 等しくないなら偽	a = b とは書かない
a != b	aとbが等しくないなら真， 等しいなら偽	a ≠ b という意味

a=b と a==b の違い

- a = b
  - 「bの値をaに代入する」という意味
  - b = a とは意味が全く違う
    - （aの値をbに代入する）
- a == b
  - 「aとbの値が等しいか比較する」という意味
  - 等しければ真，等しくなければ偽
  - b == a と書いても全く同じ

例題4.1

```
#include <stdio.h>
int main(void)
{
  int a, b;
  printf( "a = " );
  scanf( "%d", &a );
  printf( "b = " );
  scanf( "%d", &b );
  if( a == b )
    printf( "equal¥n" );
  return 0;
}
```

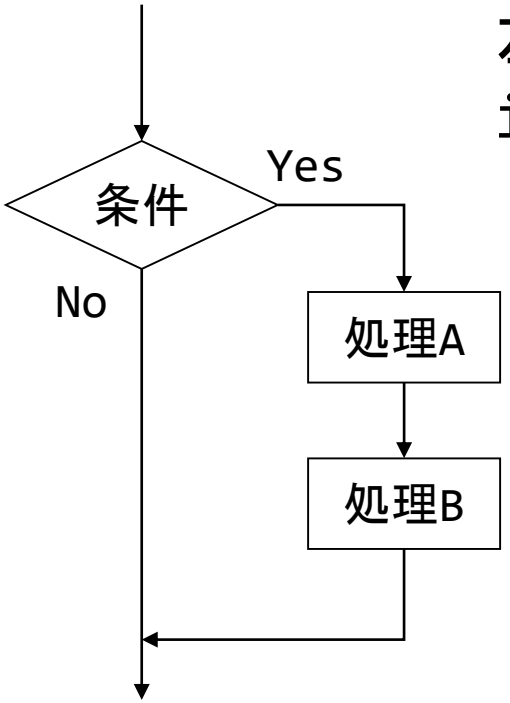
変数aに整数を代入する

変数bに整数を代入する

aとbが等しいかどうか調べる

等しければ，"equal"と表示

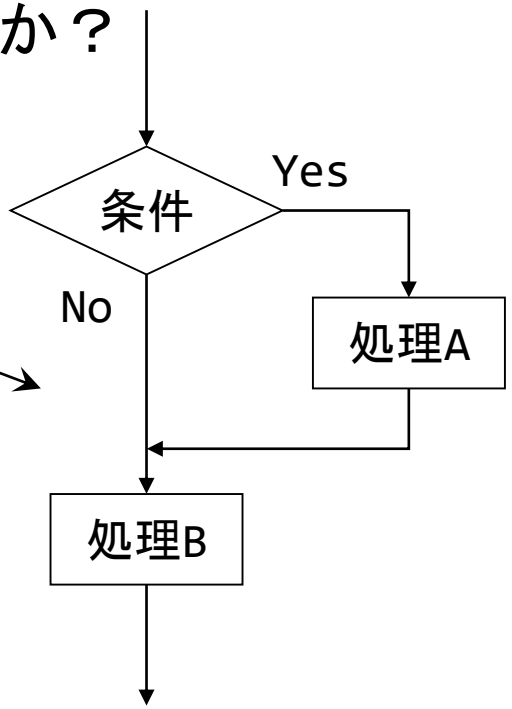
複雑なif文(1)



左のフローチャートを
if文で表すとどうなるか？

```
if( 条件 )
    処理A;
    処理B;
```

ifには
１つの文しか
書けない



このように書くと，
emacs上できれいに揃わない
⇒　そこで気がつくように！

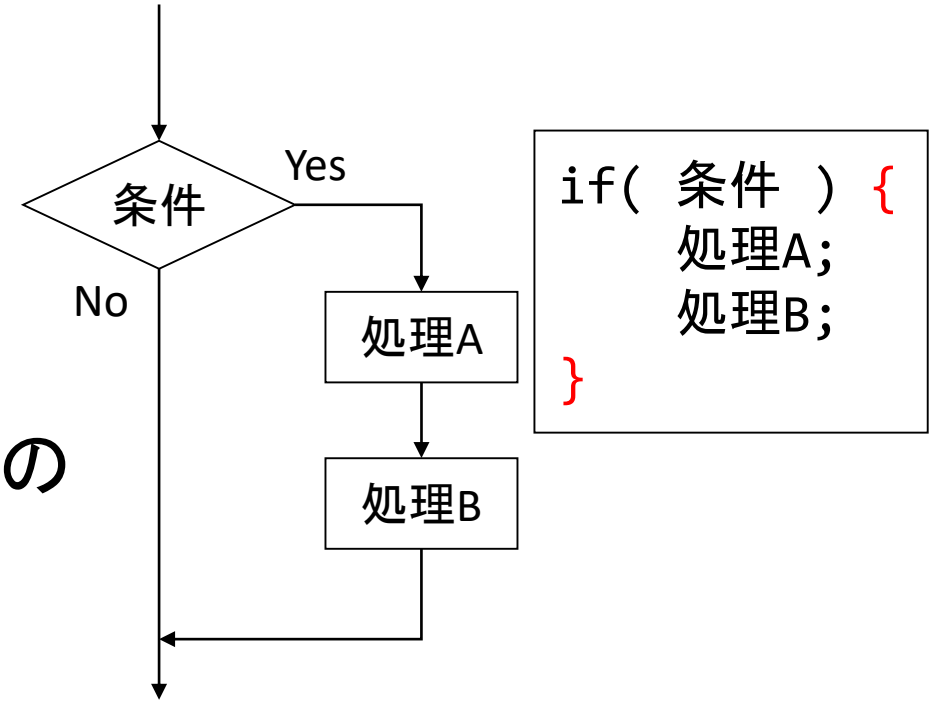
複雑なif文(2)

では，どうするか？

⇒　ブロックを使う

ブロック：

{ と } で囲んだもの



ブロックの中には，いくつ文を入れても構わない
（１つでも構わない）

ブロックの例

```
int x;
printf("正の数を入力してください：");
scanf("%d", &x);
if (x <= 0) {
  printf("x=%dです¥n", x);
  printf("これは正の数ではありません¥n");
  printf("正の数を入力してください¥n");
}
```

ブロックの後にはセミコロンは不要
ブロック内では，文末にセミコロンが必要

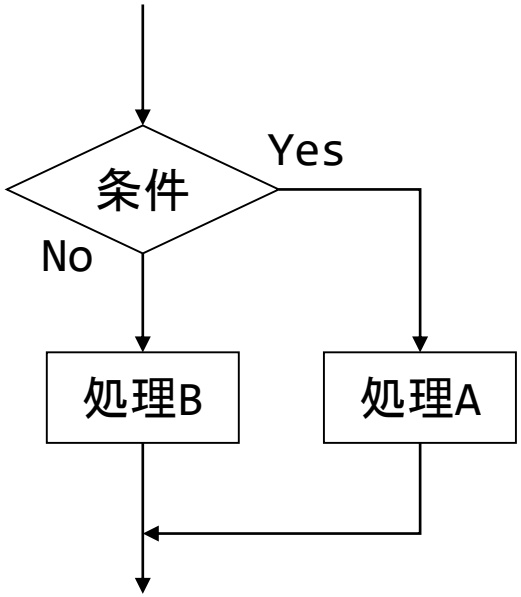
プログラミングの基礎　第2回

- 分岐処理
  - 分岐処理の復習
  - 分岐処理をC言語で表現するには？
    - if文
    - 関係演算子
    - **複雑なif文**
    - **複雑な条件式**
    - switch文
  - 教科書p.45～p.60参照

複雑なif文 (3)

- 条件が偽の場合の処理はどう書くか？

```
if ( 条件 ){
    処理A;
}
else{
    処理B;
}
```

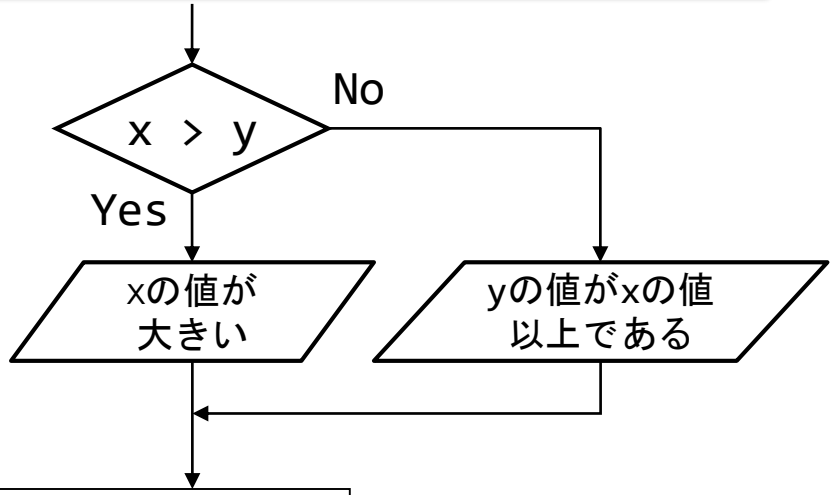


もちろん，ブロックも使える

「条件」が真なら，
「処理A」を実行
偽なら，
「処理B」を実行

if-else文の例

- ２つの変数xとy
- どちらが大きいか？




```
if (x > y){
    printf("xの値が大きい ¥n");
}
else{
    printf("yの値がxの値以上である¥n");
}
```

真/偽の両方に分かれる分岐処理があれば，
if-else文に置き換えれば良い

字下げ（インデント）

```
if( test > 40 ) {
printf("合格¥n");
} else {
printf("不合格¥n");
printf("再受験して下さい¥n");
}
```

```
if( test > 40 ) {
    printf("合格¥n");
} else {
    printf("不合格¥n");
    printf("再受験して下さい¥n");
}
```

どちらが読みやすいか？

右側のように，文の構造に合わせて書き始める位置をずらすことを「字下げ」（インデント；indent）という．
emacsではTabキーを押すと自動的に最適な位置に字下げする．
必ずTabキーを使って字下げすること

字下げのスタイル

プログラムの書き方　（これを字下げスタイルという）

```
if (test > 40) {
    .........
    .........
}
else {
    .........
    .........
}
```

```
if (test > 40) {
    .........
    .........
} else {
    .........
    .........
}
```

```
if (test > 40)
{
    .........
    .........
} else {
    .........
    .........
}
```

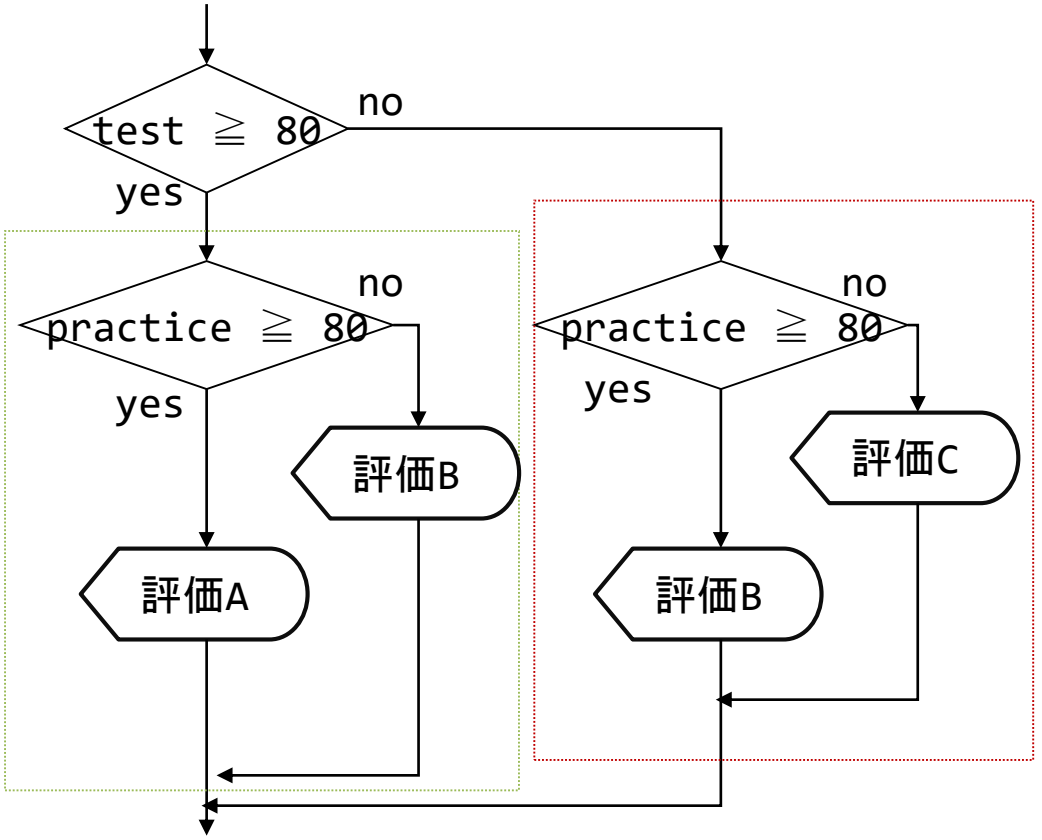
```
if (test > 40)
{
    .........
    .........
}
else
{
    .........
    .........
}
```

どの書き方でも構わない
（ただし，一つに決めたら統一すること）

if文のネスト

- if文の中にif文を入れることができる
  - ネスト （入れ子） 構造という

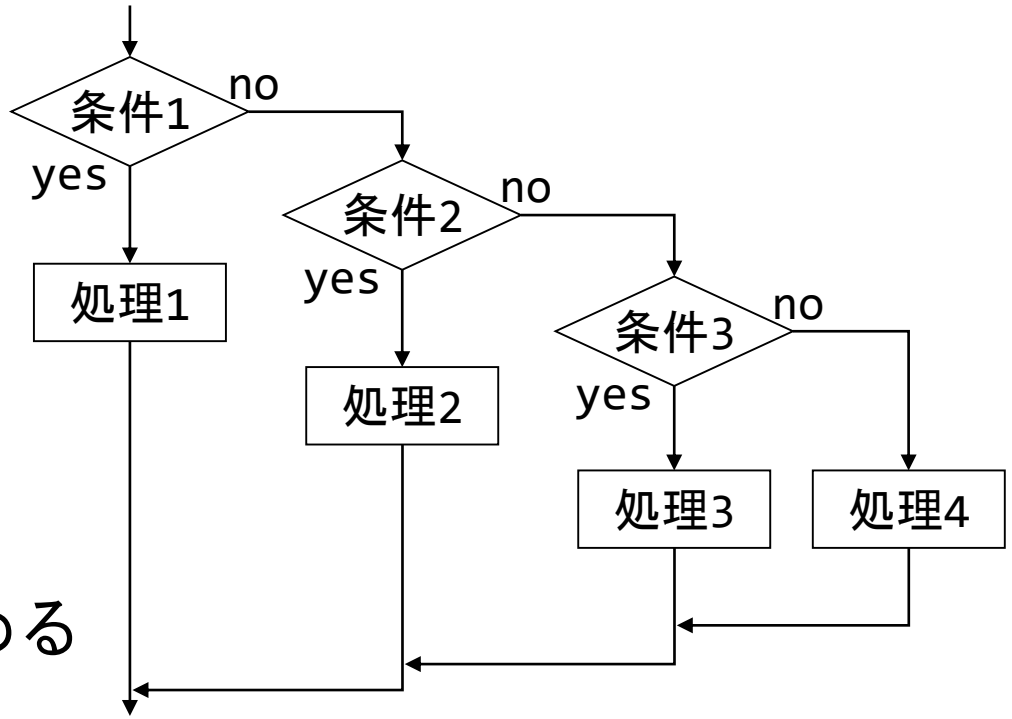
```
if (test >= 80) {
    if (practice >= 80)
        printf( "評価A¥n" );
    else
        printf( "評価B¥n" );
} else {
    if (practice >= 80)
        printf( "評価B¥n" );
    else
        printf( "評価C¥n" );
}
```



複雑なif文 (5)

- else-ifは何個でも続けることができる

```
if (条件1)
    処理1;
else if (条件2)
    処理2;
else if (条件3)
    処理3;
else
    処理4;
```

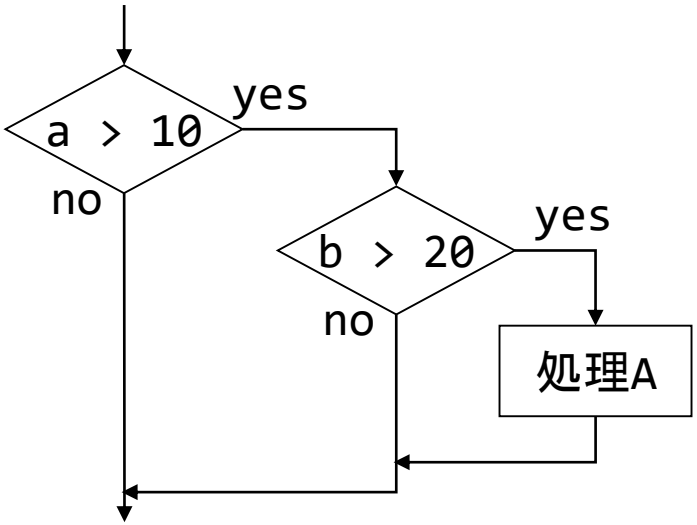


これを１つにまとめる
→ switch文（後述）

複雑な条件式 (1)

- a > 10 と b > 20 の両方が同時に真である時に，何か処理をしたい

```
if( a > 10 ) {
    if( b > 20 )
        処理A;
}
```

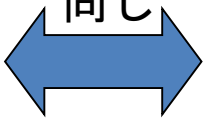


これを１つにまとめる方法がある！

複雑な条件式 (2)

- a > 10 と b > 20 の両方が同時に真である時に，何か処理をしたい

```
if ( a > 10 ) {
    if ( b > 20 )
        処理A;
}
```




```
if ( a > 10 && b > 20 ){
    処理A;
}
```

&& は，２つの条件が
両方とも真である時にYesになる

論理演算子

演算	意味	記号	例
論理積	両方とも真	&&	a > 10 && b > 20
論理和	少なくとも片方は真	||	a > 10 || b > 20
否定	真ではない時	!	! (a > 1)

論理積	a > 10	a <= 10
b > 20	Yes	No
b <= 20	No	No

a>10とb>20の両方が真
⇒　全体も真

論理和	a > 10	a <= 10
b > 20	Yes	Yes
b <= 20	Yes	No

a>10かb>20のどちらか片方（あるいは両方）が真
⇒　全体も真

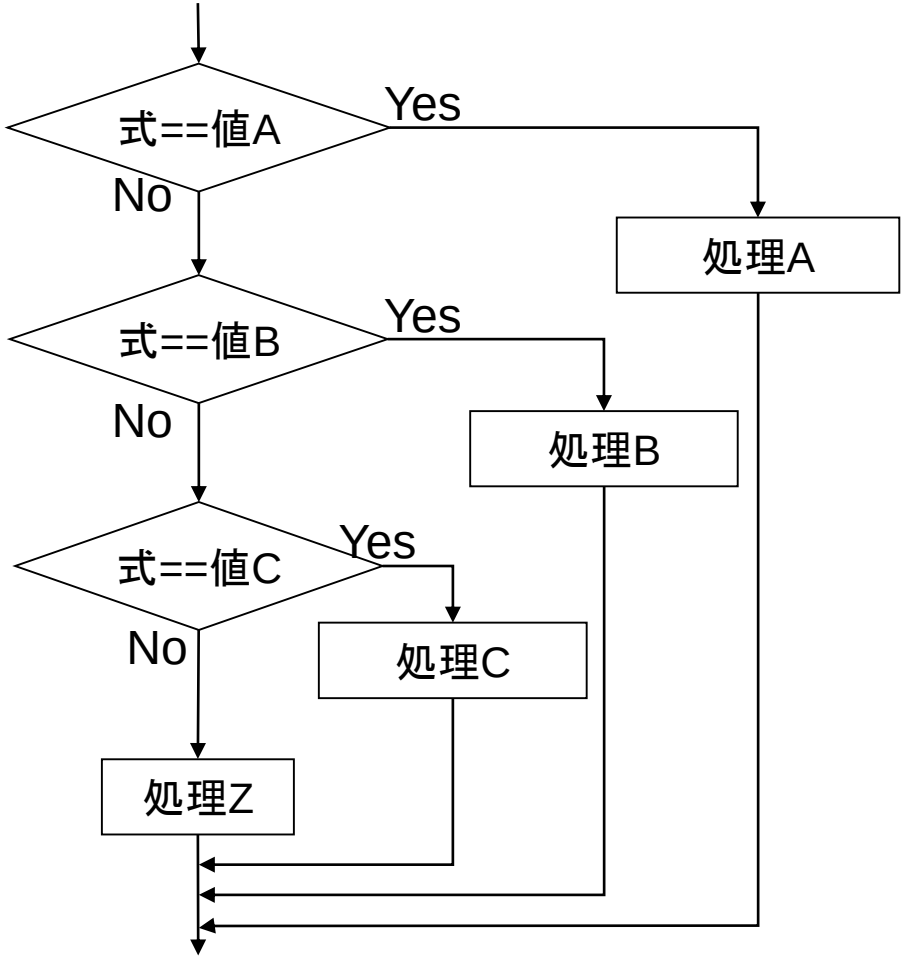
プログラミングの基礎　第2回

- 分岐処理
  - 分岐処理の復習
  - 分岐処理をC言語で表現するには？
    - if文
    - 関係演算子
    - 複雑なif文
    - 複雑な条件式
    - switch文
  - 教科書p.45～p.60参照

switch文 (1)

- 式の値によって処理を分ける
- 書式:

```
switch( 式 ) {
  case 値A:
    処理A;
    break;
  case 値B:
    処理B;
    break;
  case 値C:
    処理C;
    break;
  default:
    処理Z;
    break;
}
```



switch文 (2)

- 「式＝＝値」形式の条件のみ使える.
  - 「式＞値」などは使えない.
- case文の値の後はコロン(:)であることに注意する

```
switch( 式 ) {
  case 値A:
    処理A;
    break;
  case 値B:
    処理B;
    break;
  case 値C:
    処理C;
    break;
  default:
    処理Z;
    break;
}
```

同じ

```
if( 式 == 値A )
    処理A;
else if( 式 == 値B )
    処理B;
else if( 式 == 値C )
    処理C;
else
    処理Z;
```

switch文 (3)

```
switch( 式 ) {
    case 値A:
        処理A;
        break;
    case 値B:
        処理B;
        break;
    case 値C:
        処理C;
        break;
    default:
        処理Z;
        break;
}
```

「式の値」が「値A」と等しい時の処理

caseはいくつ書いても構わない

コロン：

セミコロン；

「値A」～「値C」以外の場合の処理

switch文の例

```
switch( i ) {
    case 1:
        printf( "case 1¥n" );
        break;
    case 2:
        printf( "case 2¥n" );
        break;
    case 3:
        printf( "case 3¥n" );
        break;
    default:
        printf( "default¥n" );
        break;
}
```

i == 1ならば,
"case 1"と表示

i == 2ならば,
"case 2"と表示

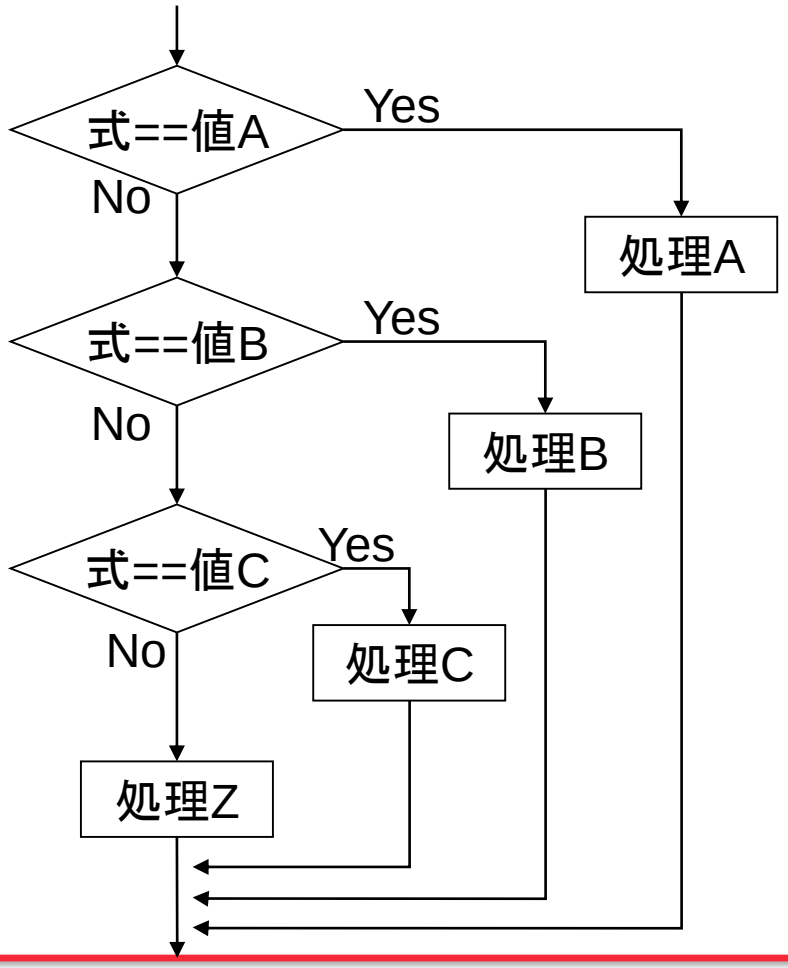
i == 3ならば,
"case 3"と表示

それ以外ならば,
"default"と表示

switch文とbreak文 (1)

```
switch( 式 ) {
    case 値A:
        処理A;
        break;
    case 値B:
        処理B;
        break;
    case 値C:
        処理C;
        break;
    default:
        処理Z;
        break;
}
```

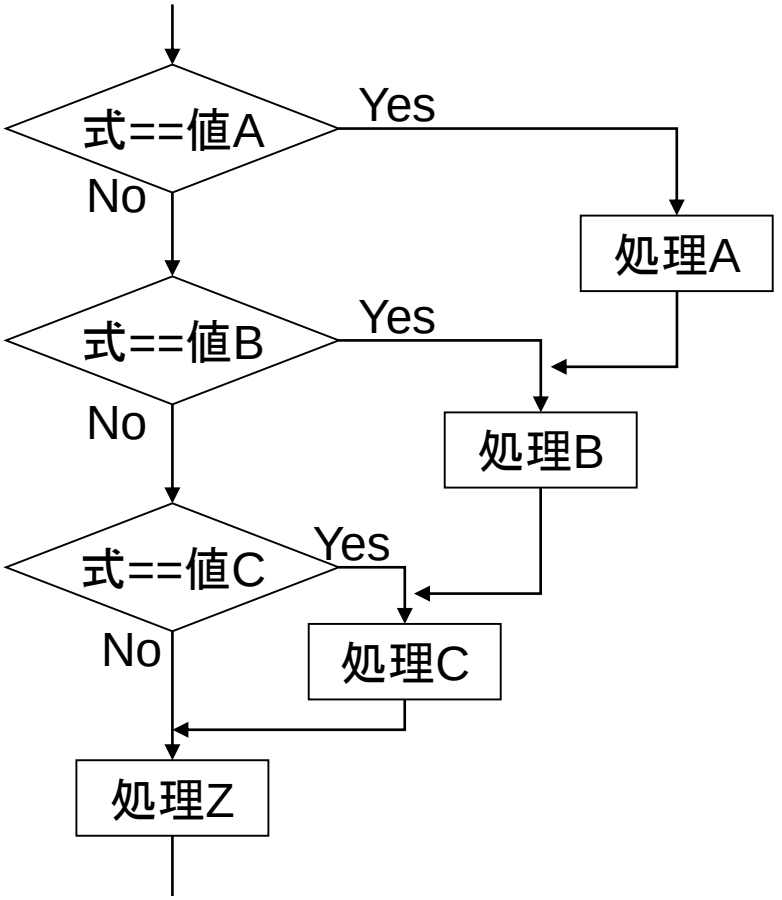
break文の書き忘れに注意する



switch文とbreak文 (2)

```
switch( 式 ) {
    case 値A:
        処理A;

    case 値B:
        処理B;

    case 値C:
        処理C;

    default:
        処理Z;

}
```

break文を書き忘れた場合



defaultのbreakはなくても同じ
しかし，習慣として書いておく方が良い

switch文とbreak文 (3)

- break文がある場合
  - switch文全体を終了する
  - 通常は全てのcaseにbreak文を付ける
- break文がない場合
  - 次のcaseを実行する
  - 故意にbreak文を省略する場合もある
    - p.59 例題4.7

switch文の例: caseを繋げる時

```
switch( i ) {
    case 1:
    case 2:
        printf("case 1 or 2¥n");
        break;
    case 3:
        printf("case 3¥n");
        break;
    default:
        printf("default¥n");
        break;
}
```

【演習課題】

- 演習課題提出システム「第2回」演習課題実施
- 演習課題2-1，演習課題2-2，演習課題2-3
  ⇒演習時間内に解く
  - 演習課題2-1
    - 演習時間の後半に答え合わせ
    - レポートに清書したものを付ける
  - 演習課題2-2，演習課題2-3
    - 設計後にプログラムを作成し，課題提出システムへ
    - 時間内に終了しない場合　⇒　宿題
- 演習課題2-4，演習課題2-5
  - 宿題（演習課題2-1～2-3が早く終了した者は授業中に）
    - 設計後にプログラムを作成し，課題提出システムへ
- 授業翌週水曜日の午後5時までに必ず提出

演習課題のヒント

- 変数に格納されている値の画面表示のフィールド幅と桁数を指定できる（教科書p.23, 表2.5）

```
x: double型
printf("x = %.3f¥n", x);
```

- これは，小数点以下３桁で表示と指定

演習課題のヒント: 変換指定子

変換指定子のフォーマット指定

- %m.nf
  - m桁分の幅が確保される
    - 小数点も全体の桁数に数える
  - 小数点以下n桁をする表示
    - 小数点以下の桁数のみを指定したい場合は，mを書かないか0とする
  - 右詰でm桁に満たない場合は，先頭に空白が表示される
    - `printf("%8.2f", -123.45);` ⇒ □-123.45
    - `printf("%.2f", -123.45);` ⇒ -123.45
    - `printf("%0.2f", -123.45);` ⇒ -123.45

【レポート】

- 演習課題提出システムの
  演習課題2-1, 2-4, 2-5について，下記内容を記載し，
  印刷し，授業時に配布した表紙をつけた，
  ”レポート”を提出せよ.
  - 演習課題2-1
    - 表を清書したものを，”レポートファイル”に貼り付け.
      （プログラム提出なし）
  - 演習課題2-4, 2-5
    - プログラム設計（フローチャート）をastahにて作成し「ツール」，「画像出力」にてPNG形式ファイルを作成し”レポートファイル”に貼り付け.
    - テスト仕様書を”レポートファイル”に貼り付け.
    - ソースコード，コンパイル結果，実行結果をscriptコマンドでファイルにし印刷し”レポートファイル”に貼り付け.
- 授業翌週水曜日の午後5時までに
  55号館304室前のレポート提出小箱へ提出

レポートの書き方

- レポートで重要なこと
  - 「読んだ人が理解できるか」
  - 「読んだ人に伝わるか」
  - 「書いた人が理解しているか」は重要ではない
  - 「書いた本人」が理解していても，
    「読む人」に通じなければ意味がない

筆記課題（レポート）執筆時の注意点

- セクション番号，タイトルを付ける
- フォントの大きさ，種類を適切に
- フローチャートは見やすく，
  分岐にはYes/Noを必ず書く
- ページ番号をフッターに付ける

他にも様々な点があります．
レポートの書き方の本を読むことを推奨します．

テスト

- プログラムが意図した通りに
  動作しているか否かを調べる必要がある
  - すべての入力を試すことは現実的に不可能
  - 効率よくテストケースを作成する必要がある
- 実際に行ったテストについて表形式で記述
  - テスト番号
  - テスト項目
  - 入力
  - 期待される出力・動作
  - テスト結果（日付）

同値分割（1/2）

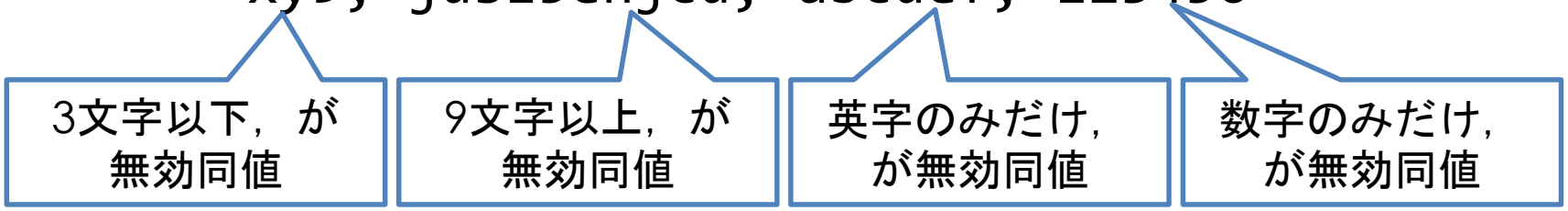
- 同値分割：同値クラスの識別
  - 機能仕様の入力条件を満足する範囲（有効同値クラス）と満足しないクラス（無効同値クラス）に分割することでテストケースを作成
  - 機能仕様の例
    1. パスワード（入力条件：文字数4～8，且つ，英字と数字の組み合わせ）を設定する
    2. 入力値が入力条件を満たすかでOK/NG表示を分岐

入力条件	有効同値クラス	無効同値クラス
文字数	4～8	3以下，9以上
文字の種類	英字と数字の組合せ	英字のみ，数字のみ

同値分割（2/2）

クラスに基づくテストケースの作成

- 有効同値クラスを検査するテストケースを作成
  - Amku5ge
- 1つの無効同値クラス（無効同値クラスが重ならないように）と残りの有効同値クラスを検査するテストケースを作成
  - xy9, jdsi5enjcd, abcdef, 123456



境界値分析（限界値分析）

- 境界値分析
  - 入出力条件の境界値を詳しくテストする
    テストケースを作成

条件	1～64の数字
境界	1と64

1. 入出力条件の識別
   - 機能仕様の入出力条件に着目し、境界を判別する
     （機能仕様例：1～64の数字を入力する．．．）
2. 両方の境目になる値（ONポイント），
   その一つ外になる値（OFFポイント）選択
   - 0,1,64,65

offポイント　1と64はonポイント　offポイント

テスト仕様書

- 入力値範囲の条件が1～64の仕様で，
  範囲外の場合”範囲外”と表示し，
  範囲内の場合は入力値を表示する
  プログラムの場合

テスト番号	テスト項目	入力	期待される出力	テスト結果
1	下限の境界(off)	0	範囲外	10/1
2	下限の境界(on)	1	1	10/1
3	上限の境界(on)	64	64	10/1
4	上限の境界(off)	65	範囲外	10/1

ここまでは設計終了後に作成

プログラム作成後にテストし，
期待通りなら実施日を記入