プログラミング基礎
第5回

藤江 真也

2021年5月21日

課題の再提出について

- 第3回，第4回の課題の成績を公開しました
- 未提出や再提出の人は期限が
  5月25日23時59分
  に設定されていますのでそれまでに提出をしてください

講義・課題に関する質問について

- 講義用Slackの各グループのチャンネルに質問をしてください
- みんなに見えるというのは抵抗があるかもしれませんが…
  - 過去の情報が参考になるかも
  - 自分の情報が他の人の役に立つかも
  - ダイレクトメッセージだと私にしか見えませんのでTAその他からのヘルプが受けられません

講義・課題に関する質問について

- 多い質問
  - プログラミング環境
  - コンパイルエラー
  - ファイル操作
- このような質問も歓迎ですが…
- プログラミングに関する質問もどしどしください

例えば以前にあった例

質　問　①

intとfloatの活用する場面や意味がわかりません。
例えなどを入れて、専門用語など少なめで初心者でもわかりやすいように教えてください。
整数型や実数型もイメージつきません。

- なぜ整数(int)と実数(float)があるのか
  - 整数の2は，実数(小数)の2.00000… と同じなんだから整数をわざわざ用意するメリットはないじゃないか！？という疑問はもっとも…

- 実数があれば十分か？
  - これまで勉強してきた数学ではYESとも言える（あらゆる整数は実数でもあるので…）
- しかし，**コンピュータの世界はデジタル**なので，厳密な意味では整数しか扱えない
- 実数に見えるものも整数を用いて**近似的**に扱っているので，float や double 型の変数に入っている値は**誤差**を考慮して扱う必要がある（本日の課題1で明らかになる）
- 整数だけで済む問題は整数だけで解いた方がよい（誤差を考える必要がない）

質　問　②

scanf("%d",&x)のxの前についてる&ってどういう意味ですか？
あと%dではなく%4dの時は何が違うのでしょうか。
よろしくお願いします。

- &については，ポインタというものの勉強が必要です．後日説明するのでそれまでは「scanfの場合は必要」とだけ覚えておいてください．
- %4dの4は，「4ケタ未満でも4ケタで表示する」という意味になります．
  - 32を%dで表示すると「32」，%4dで表示すると「□□32」(□は半角スペース)となります．

ファイルの
コピー，移動，削除

ファイルのコピー，移動，削除

- プログラムのソースコードのバックアップをとるなどの理由でファイルをコピー（複製）したりするとよい
- コピー，移動，削除の方法を紹介する

- 基本的な操作であるが故に事故がおこりやすい
  - どの操作も誤操作によりファイルが消えるリスクあり
  - 特に理由がなければ**移動や削除はしない方がよい**

ファイルのコピー（複製）... cpコマンド

```
$ ls
pingpong.c
$ cp pingpong.c pingpong_1.c
$ ls
pingpong.c      pingpong_1.c
```

cpコマンドで，
pingpong.c を pingpong_1.c
という名前で複製する
（もともと pingpong_1.c が
無ければ新規作成される）

gedit などで pingpong.c を上書き保存したあと

```
$ ls
pingpong.c      pingpong_1.c
$ cp pingpong.c pingpong_1.c
$ ls
pingpong.c      pingpong_1.c
```

pingpong_1.c が新らしい
pinpong.cで上書きされる
（**古いpingpong_1.cは失われる**）

※cpコマンドでディレクトリをコピーするにはオプションが必要

ファイルの移動（名前変更）...mvコマンド

```
$ ls
pingpong.c
$ mv pingpong.c pingpong_1.c
$ ls
pingpong_1.c
$ mkdir backup
$ ls
backup          pingpong_1.c
$ mv pingpong_1.c backup/
$ ls
backup
$ mv backup bk
$ ls
bk
```

mvコマンドで，
pingpong.c を pingpong_1.c
という名前に変更する
（**もともと pingpong.c は
失われる**）

mvコマンドで，
pingpong_1.c をディレクトリ
backupに移動する

mvコマンドは，
ディレクトリの移動（名前変更）
もできる

ファイルの削除 ... rmコマンド

```
$ ls
pingpong.c
$ rm pingpong.c
$ ls
$
```

rmコマンドで，**pingpong.c を削除**

※rmコマンドでディレクトリを削除するにはオプションが必要

本日の内容

- 条件分岐
  - if文
  - 比較演算子
  - 論理演算子

準　備

- ファイル(0521.tgz)をダウンロード

```
$ wget http://sites.fujielab.org/ip/files/0521.tgz
```

- ダウンロードしたファイルを展開

```
$ tar zxvf 0521.tgz
```

- 展開されたディレクトリに移動

```
$ cd 0521
```

- if.cなどのファイルがあることを確認

```
$ ls
```

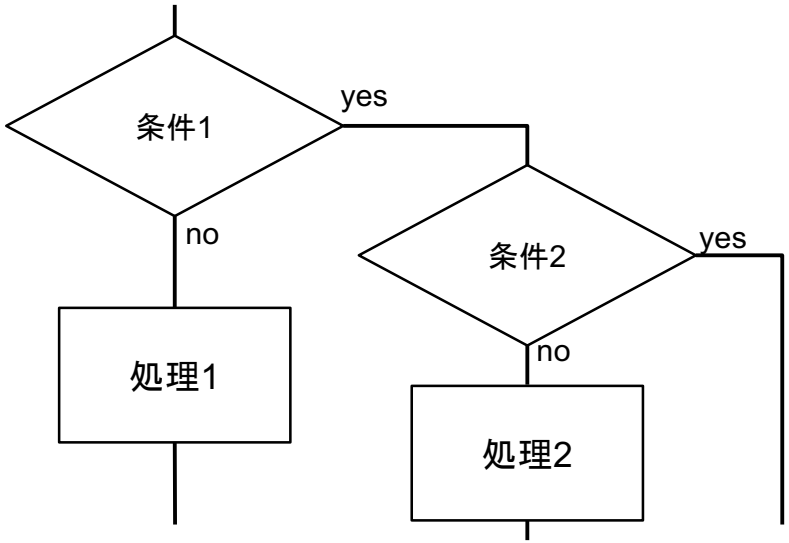

講義の進め方

- サンプルプログラムを実行
  - 今回からは資料内にはファイル名のみを書きます
- プログラム内のコメントを見ながら説明
  - /* */ で囲まれている部分
- スライドで補足説明
  - スライド(講義資料)は手元で確認できるようにしておくとよい
- サンプルプログラム内の練習で理解をして理解を深める

条件分岐

(参考書 第3章 プログラムの流れの分岐)

フローチャートではひし形の記号



文にはどんなものがあるか？

- 変数宣言文（variable declaration statement）
- 代入文（assignment statement）
- 関数呼び出し文（function calling statement）
- if文（if statement）
- switch文（switch statement）
- for文（for statement）
- while文（while statement）

などなど

if 文

odd.c

条件式(condition expression)

```
if (x % 2 == 1) {
    printf("%4dは、奇数¥n", a);
}
```

ブロック

if文では，条件式に指定した式が成立するときに限り，
次に続くブロックが実行される

真（true / yes）... 条件が成立すること
偽（false / no）... 条件が成立しないこと

奇数の判定

```
#include <stdio.h>

int main (void) {
  int x = 30;
  if (x % 2 == 1) {
    printf("%4dは奇数¥n", x);
  }
  return 0;
}
```

スペースの関係で例示するプログラムは
変数を定数で初期化しているが，
実際にはscanfで入力させた方が面白い．

偶数のときには偶数と表示するにはどうすればいいか？

if-else 文

if.c

```
if ( 条件式 ) {
    ...        ブロック A
} else {
    ...        ブロック B
}
```

条件式が真のときにのみブロックAが実行される
そうでないときにのみブロックBが実行される

if-else文

```
if ( 条件式1 ) {
    ...        ブロック A
} else if( 条件式2 ) {
    ...        ブロック B
} else {
    ...        ブロック C
}
```

条件式1が真のときにのみブロックAが実行される
そうでなくて条件式2が真のときのみブロックBが実行される
更にそうでないときにのみブロックCが実行される

条件式

rel.c

- 等値演算子（equality operators）
  - a == b ... a と b が等しい
    - = を1つにしてしまうと意味が変わるので注意!!!
  - a != b ... a と b が等しくない
- 関係演算子（relational operators）
  - a > b ... a が b より大きい（a is greater than b）
  - a < b ... a が b より小さい（a is less than b）
  - a >= b ... a が b 以上（a is not less than b）
  - a <= b ... a が b 以下（a is not greater than b）

条件の組み合わせ

- 条件を組み合わせたいことがある
  例1）3の倍数 か 4の倍数
  例2）3の倍数 かつ 4の倍数
- 論理演算子（logical operators）を使って
  条件式を組み合わせて1つの条件式にできる

論理演算子

logic.c

- 論理演算子(logical operators)
  - && ... 論理積演算(AND演算)

    条件式1 && 条件式2

    条件式1と条件式2の両方が真のときのみ真
  - || ... 論理和演算(OR演算)

    条件式1 || 条件式2

    条件式1と条件式2のどちらかが真なら真
  - ! ... 否定演算(NOT演算)

    ! 条件式

    条件式が偽であれば真

もう一つの条件分岐
switch文

文にはどんなものがあるか？

- 変数宣言文(variable declaration statement)
- 代入文(assignment statement)
- 関数呼び出し文(function calling statement)
- if文(if statement)
- switch文(switch statement)
- for文(for statement)
- while文(while statement)

などなど

switch文

```
switch ( 式 ) {
  case 定数式1 :
     文1;
     文2;
     break;
  case 定数式2 :
     ...
     break;
  default:
     ...
     break;
}
```

式の値が定数式1と同じ場合のみこの部分が実行される

式の値が定数式2と同じ場合のみこの部分が実行される

それ以外の場合この部分が実行される

switch文

```
switch ( 式 ) {
  case 定数式1 :
     文1;
     文2;
     break;
  case 定数式2 :
     ...
     break;
  default:
    ...
    break;
}
```

条件式ではなくて式

定数式なので，
変数は使えない

breakでブロックを抜ける
（無いとどうなるだろうか…？）

defaultが必要
（無いとどうなるだろうか…？）