# ソフトウェア概論 A/B

-- FILE 入出力/模擬試験 --

数学科 栗野 俊一 / 渡辺 俊一 ( TA: 栗原 望 / 小嶋 仁子 [M2] )

2019/01/18 ソフトウェア概論

# 伝言

## 私語は慎むように !!

- 出席パスワード : 20190118
- 色々なお知らせについて
  - 栗野の Web Page に注意する事

    http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino
- 廊下側の一列は遅刻者専用です(早く来た人は座らない)
- 講義開始前に済ませておく事
  - PC の電源を入れておく
  - ネットワークに接続しておく
  - 今日の資料に目を通しておく
- 講義前の注意
  - 講義前は、栗野は準備で忙しいので TA を捕まえてください
- やる気のある方へ
  - 今日の資料は、すでに上っています
    - どんどん、先に進んでかまいません

# 今後の予定

- 今後の予定(後ろから)
  - 2018/01/25 (講議最終日)
    - 試験を行う
  - 2018/01/18 (本日/講議最終日前)
    - FILE 入出力/模擬試験

# 前回(2019/01/11)の内容

- 前回 (2019/01/11) の復習
  - ポインター値：メモリモデルのアドレスの抽象化
    - ポインター値 = アドレス値 + 型情報
  - アドレス値 = 変数の割り当てられているメモリのアドレス値
    - 実行時の「値」は、「アドレス値」だけ
  - 型情報 = 領域サイズ + データの扱い方(演算方法/表現)
    - 領域サイズは sizeof で得る事もできる
  - ポインター値の演算
    - 整数値の加減算：+1 するとアドレス値が領域サイズだけ増える
    - キャスト：型情報を変更する事ができる

# お知らせ

- 本日の予定
  - FILE 入出力/模擬試験
- 本日の目標
  - 演習
    - 課題の提出

# 先週 (2019/01/11) の課題

- □ 先週 (2019/01/11) の課題
  - ○ 課題 先週-01:
    - ▹ ファイル名 : 先週-01-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 :「三角形の形」をした配列
  - ○ 課題 先週-02:
    - ▹ ファイル名 : 先週-02-QQQQ.c (QQQQ は学生番号)
    - ▹ 内容 : 動的なメモリの利用
- □ ※
  - ○ ファイル形式は、いずれもテキストファイル(C 言語プログラムファイル)

# 今週 (2019/01/18) の課題

- 今週 (2019/01/18) の課題
  - 模擬試験の結果を提出する
    - 複数ファイルの提出になるので、注意
- ※
  - ファイル形式は、いずれもテキストファイル(C 言語プログラムファイル)

# 次週の予定

- 2019/01/25 (次週/講議最終日)
  - 試験を行います ( 試験時間は、90 分 ～ )
    - 試験開始 30 分後に出席を取ります
  - 時間内に結果を CST Portal に提出してください
    - ギリギリに提出しようとするとサーバーが対応できない可能性あり
- 試験の形式
  - ファイルに入った問題をダウンロード
    - ファイル内に個々に問題が入っているので、それを見て解く
    - 問題は、各自異なる(ので答も異なる) / 回答は、ファイルの形で、ポータルに upload する
  - 持ち込み : Note-PC を含め、何んでも可 (もう一台の PC / 本 / ノート..)
  - 禁止事項
    - 音を出しては駄目 ( 会話不可 / 携帯電話不可 / チャット可 ) / 物の貸し借りは駄目
- 質問
  - 問題が「変」と思ったら、手を挙げてください

# 試験の開始

- 試験の開始手順
  - 問題ファイル (QQQQ.zip) を Web よりダウンロード
    - QQQQ は自分の学籍番号 (今回は 9999.zip)
  - 問題ファイルを展開 (展開先はどこでも OK)
  - question フォルダに回答/問題ファイルがある
    - 解答ファイル : QQQQ-a.txt
    - 問題ファイル : q で始まるファイル(色々ある)
  - q.txt をサクラエディタで開いて読む
    - 問題の詳しい内容は、更に別のファイルに書いてある
    - それぞれの問題を解き、解答する

# 解答の提出

- 問題の解答の提出形式は次の二通り
  - q.00 ～ q.11, q.15, q.17 の解答：一問毎にそれぞれ一ファイルを解答として提出
    - それぞれ、QQQQ-a00.c ～ QQQQ-a11.c, QQQQ-a15.c, QQQQ-a17.c として提出
  - それ以外の解答：QQQQ-a.txt にまとめて書きこむ
    - QQQQ-a.txt に答を記入し、提出
    - 基本は PC の表示をコピー・ペーストして欲しい
    - 英数字や、半角で入力できる記号は、半角で入力する
    - 逆に半角カナは利用しない
  - 最大 15 個のファイルを CST Portal に提出する
    - できた分だけ提出すればよい(提出されている分だけ採点対象になる)
    - QQQQ-a.txt も解けた答だけ記入すればよい(全て記入しなくてもよい)
  - 時間内に提出しないと提出できなくなる !!
- 試験の出欠：次の何れかで出席扱いになる
  - 試験開始 30 分に出席カードを配布するので、それに記入・提出する
  - CST Potal に、一つでもファイルが提出できている

# 模擬試験と本番の違い

| 項目 | 模擬テスト | 本 |
|---|---|---|
| 開始時間 | 既に参照可能 | 試験開始 |
| 提出期間 | 時間内 (後に翌週まで延長) | 時間内 (時間切 |
| 問題 | 全員共通 | 各人異 |
| 前半と後半 | 共通 | 問題の一 |
| 問題ファイル | 20180717-9999.zip | 20180724-QQQQ.zip |
| QQQQ-a.txt | 9999-a.txt の名前を変更 | |
| 会話 | 可能 (おおいに相談しよう) | 不可 (無言で |
| 質問 | 可能 (相談する最後の機会) | 不可 (問題の不備 |
| PC 対応 | 可能 (相談する最後の機会) | 不可 (一切 |

# 本日の目標

- □ 本日の目標
  - ○ 本番の試験に備える
    - ▷ 環境は大丈夫か？( Note-PC / Soft / Network .. )
    - ▷ 課題は解けるか？( 模擬試験の問題 / 過去の課題 .. )
  - ○ PC の「ヘルス証明」
    - ▷ トラブルが解っている人：申し出る / 今日の午後に解決できれば..
    - ▷ トラブルのない人：課題を提出する / 今週迄は「健康だった」という事
  - ○ 「ヘルス証明」ができている人は、来週トラブルがあっても配慮する
    - ▷ 上記の両方ができていない人は、来週、PC トラブルがあっても配慮しない
    - ▷ cf. ド＝モルガン
  - ○ ポイント
    - ▷ 「資料を見ながら、作業ができる」ようにする事
- □ 試験日について
  - ○ 体の容態が悪いならば、無理せず、メールして休む
    - ▷ 試験日は、相談して、翌週以降にやりましょう..

# 動的なデータ構造

- 静的なデータ構造
  - 配列/単純変数：保存できるデータのサイズは固定
    - alloc を使っても、配列ならば、その中に入るデータの個数は固定
    - メモリの無駄を覚悟すれば、「大きなサイズの配列」でも代用可
- 動的なデータ構造
  - 保持できるデータ数が可変長(本質的に動的)
    - alloc を利用しないと扱えない
- 動的なデータ構造の例
  - List (リスト)：複数の要素を「並べた」もの

# 構造体へのポインター値

- 構造体のタグ
  - 構造体には、構造体を区別する名前(tag:タグ:構造体名)がつけられる
    - struct タグ { .. }
  - 「sturct タグ」は構造体型の名前として利用できる
    - これまでは、これを typedef でサボッていた
  - ポインタによる自己参照を行う場合は、タグがどうしても必要になる
- 構造体へのポインタ
  - 構造体へのポインタを利用して、構造体の要素にアクセスできる
    - p が、要素 x を持つ構造体へのポインタなら (*p).x で参照で可能
    - 構造体へのポインタの要素の参照はよくあるので、省略記号がある
    - 「(*p).x」の代りに「p -> x 」という表現が利用できる

# ファイル I/O

- ファイル操作
  - ファイル：データを恒常的記憶するための仕組み
    - メモリの内容は、PC の電源を切ると失われる
  - リダイレクション(復習)
    - 「<」、「>」を利用して、プログラムの入出力をファイルに変更可能
    - せいぜい、入力と出力に一つずつしかファイルが利用できない
  - ファイル I/O ライブラリ
    - 任意のファイルに対する読み(Read/Input)書き(Write/Output)する機能

# FILE 構造体

- FILE 構造体による File I/O
  - fopen 関数を利用して、ファイル I/O を管理する FILE 構造体へのポインタ値を得る
    - FILE 構造体は、「どのファイルの何処を参照しているか」を管理する情報
    - FILE 構造体を経由して、少しずつファイルからデータを得る事が可能
    - open に失敗した場合、fopen 関数は NULL を返す
  - 利用が終った FILE 構造体は fclose で必ず close する
    - 特に write 時には close しないとデータが失わる(保存されない)可能性が生じる
    - read の場合も、close しないと「資源の無駄使い」になる
    - cf. memory の alloc/free と同様な関係
  - データの入出力
    - fgetc/fputc : 文字単位の File I/O
    - fscanf/fprintf : ファイルに対する printf/scanf
    - fread/fwrite : ファイルに対する固定長のデータの I/O