

ソフトウェア概論 A/B

-- データ構造 (8) --

(FILE 入出力)

数学科 栗野 俊一 / 渡辺 俊一

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

2020/12/18 ソフトウェア概

伝言

□ 出席パスワード : 20201218

□ 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>

□ 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)

○ 2021/01/22 (講義最終日)

▶ 試験を行う : リアルタイム講義

○ 2021/01/15 (講義最終日前)

▶ 模擬試験を行う / 落穂拾い : リアルタイム講義

○ 2020/12/25, 2021/01/01, 2021/01/08

▶ 冬期休暇期間中 : この講義はない

○ 2020/12/18 (本日)

▶ データ構造 (8) : FILE 入出力

前回の内容

ソフトウェア概論 A/B (2020/12/18)

前回の内容

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

前回の内容

□ 前回の内容：データ構造 (7)

○ 講義

- ▶ 色々な型と型変換：型のサイズ/型の昇格/キャスト
- ▶ 動的メモリの確保：malloc/free

本日(2020/12/18)の予定

ソフトウェア概論 A/B (2020/12/18)

本日(2020/12/18)の予定

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

本日(2020/12/18)の予定

□ 本日(2020/12/18)の予定

○ データ構造 (8)

▶ FILE 入出力

□ 本日の目標

○ 演習

▶ 課題の提出

課題

ソフトウェア概論 A/B (2020/12/18)

課題

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

今週 (2020/12/18) の課題

□ 今週 (2020/12/18) の課題

○ 課題 20201218-01:

- ▶ ファイル名 : 20201218-01-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 整数値のキュー(queue)

○ 課題 20201218-02:

- ▶ ファイル名 : 20201218-02-QQQQ.c (QQQQ は学生番号)
- ▶ 内容 : 二つのファイルを比較して最初に異なる場所を表示する

□ ※

- 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)

先週 (2020/12/11) の課題

□ 先週 (2020/12/11) の課題

○ 課題 20201211-01: float と double の精度の差

▶ ファイル名 : 20201211-01-QQQQ.c (QQQQ は学生番号)

▶ 内容 : 自然対数の底の計算

○ 課題 20201211-02:

▶ ファイル名 : 20201211-02-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)

動的なデータ構造

ソフトウェア概論 A/B (2020/12/18)

動的なデータ構造

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

void 型と NULL ポインター

□ void 型

- void : 関数が値を返さない時の宣言型

- ▶ 「引数がない」場合にも、「引数宣言で『void』とする」

□ void へのポインタ型 (void *)

- どの型へのポインタ値かが不明な場合のポインタ値を表す型

- ▶ 基本は「意味(型)がない」ポインタ値 ((void *)型に * を付けても値が取り出せない)

- NULL ポインタ : 「無意味なポインタ値」を表すポインタ値

- ▶ 「意味のあるポインタ値と異なる事」が保証されている (「#define NULL ((void *)0)」と定義されている)

- ▶ ポインタ値を返す関数のエラーが生じた時の値として用いられる

- ▶ 例 : malloc : (メモリ確保) に失敗した場合 / fgets(入力) : 入力に失敗

動的なデータ構造

- 静的なデータ構造 (保持できるデータサイズを最初に決める)
 - 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 (保持できるデータサイズを後から決める)
 - 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 { .. }`
- 「`struct 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` でサボっていた
- ポインタによる自己参照を行う場合は、タグ名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`」の代わりに「`p -> x`」という表現が利用できる

ファイル I/O

ソフトウェア概論 A/B (2020/12/18)

ファイル I/O

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

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れる(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/printf` : 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

大域変数

ソフトウェア概論 A/B (2020/12/18)

大域変数

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます

大域変数：変数の有効範囲(scope)と有効期間(lifetime)

□ 大域(グローバル)変数

○ 大域変数とは：複数の関数で共有可能な変数 (cf. 局所変数：ブロック内のみ有効)

▶ 構文：関数の外で変数宣言する (cf. 局所変数は、ブロック内部)

▶ 意味：プログラムの開始時(プログラムロード時)に初期化/ずっと有効

○ ファイル間での大域変数の共有

▶ 一つのファイルで、変数宣言(定義:一度だけ)。他では `extern` 宣言する

□ 変数の有効範囲(scope)

○ 同じ変数名が、(ソースコード内の)何所から何処までが同じ領域を指すか

▶ コンパイル時の概念/空間

□ 変数の有効期間(lifetime)

○ 同じ変数名が、(プログラム実行内の)何時から何時までが同じ領域を指すか

▶ 実行時の概念/時間

標準ファイルポインター

□ 標準ファイルポインター

- 標準ファイルポインターは、プログラム実行開始(main 関数の呼出し前)に作られる
 - ▶ 大域変数の形で提供される
- 標準ファイルポインター
 - ▶ stdin : 標準入力
 - ▶ stdout : 標準出力
 - ▶ stderr : 標準入力

おしまい

ソフトウェア概論 A/B (2020/12/18)

おしまい

講義内容の静止画・動画での撮影、及び SNS 等への転載を固く禁じます