ソフトウェア概論 A/B

-- 動的データ構造 --

数学科 栗野 俊一

2011/12/09 ソフトウェア概論

伝言

私語は慎むように !!

- 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)
- 講義開始前にすませておくこと
  - PC の電源を入れる
  - ネットワークに接続しておくこと
  - 今日の資料に目を通しておくこと
- 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

前回の復習：printf/scanf

- printf (char *, ...) : print with format
  - 数値を書式(format)付きで出力するライブラリ関数
    - ex) printf ("a=%d, b=%f\n", 123, 9.87); →「a=123, b=9.870000(改行)」
  - 一つ目の引数で指定した文字列の中に数値を埋め込んだ形の出力が可能
    - 文字列の中の数値に対応する場所には、書式指定(「%～」)がされる
    - 書式指定されている部分には、二つ目以後の数値が埋め込まれる
  - 書式指定の例
    - %d -- 整数, %c -- 文字, %s -- 文字列, %f -- 浮動小数点数
- scanf : scan with format
  - 数値を書式(format)付きで入力するライブラリ関数
    - 書式の形式は、printf とほぼ同じ (double の時は %lf を使う)
    - 二つ目以後の引数は、入力データを保存する場所のポインター値
    - 文字列を入力する場合 (%s) の時も当然ポインター値だが、その場合は配列の先頭アドレスを指定

お知らせ

- 本日の予定
  - 動的データ構造
  - C 言語でオセロ(2)
- 本日の目標
  - 演習
    - 課題の提出

前回 (2011/12/02) の課題

- 前回 (2011/12/02) の課題
  - 課題 1: [sample-006.c 参照]
    - ファイル名 : 20111202-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : myprintf で Point2D 型を出力する %D が扱えるように拡張する
    - ファイル形式 : テキストファイル(C 言語プログラムファイル)

今週 (2011/12/09) の課題

- 今週 (2011/12/09) の課題
  - 課題 1: [sample-001.c 参照]
    - ファイル名 : 20111209-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : IList 内の指定したデータが何番目かを表示する関数を作成する
    - ファイル形式 : 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

- □ データ型とコーディング
  - ○ コーディング：現実にある「物」を表現する仕組み
  - ○ データ型：「物」の集合
    - ▹ 表現方法 (型宣言) と、操作方法 (演算:プログラム) の対で表現
  - ○ データサイズの問題
    - ▹ 固定なサイズのデータ型だけで、現実にある「物」が表現できるか？
    - ▹ 基本は Yes (整数/文字列..) だけど .. 効率が悪い
    - ▹ 必要なサイズを必要なだけ用意する仕組み → 動的なデータ構造
- □ 静的データ構造 (プログラム作成時に大きさが定まる)
  - ○ 基本型 (int/double/etc..)
  - ○ 導出型 (struct, 配列, ポインター 型)
    - ▹ その型のデータの大きさ(メモリサイズ)は固定：静的に表現可能
    - ▹ プログラム内で型の作成と宣言だけで基本操作が可能
- □ 動的データ構造 (プログラム実行時に大きさが定まる)
  - ○ 大きさが可変のデータ
    - ▹ alloc/free を利用する / 操作プログラムが必須

List

- List とは
  - 同じ型の要素の並び
    - その意味で配列とよく似ているが..
  - サイズは可変長 (長くなったり短くなったりする)
    - 並びの途中で要素の追加、削除ができる
- 連結 List の実装 (片方向リスト)
  - データ構造 (icell.h/icell.c/ilist.h/ilist.h)
    - List ::= NULL | 先頭の Cell へのポインター値
    - Cell ::= next (次の Cell) data (その Cell が持つデータの値)
- List の操作
  - 機能
    - 新規作成(new)/開放(free)/出力(print)/挿入(insert)/追加(append)/検索(search)
  - [ポイント] データ構造は、型宣言 + 操作関数で決る

分割コンパイル

- 分割統治法 (復習)
  - 大きな問題は、小く分割して、個別に解く (個別撃破)
    - 個々の解をまとめて、元の解を構成する (解の構成)
- 分割コンパイル
  - 大きなプログラムを小さなプログラムの集まりで表現
    - 個々にコンパイルする (分割コンパイル)
    - 個々のオブジェクトをまとめて一つの実行ファイルを作る(リンク)
  - プログラムが分れていると、矛盾する可能性がある
    - ヘッダーファイルを経由して、矛盾がないようにチェックする

sample-001.c の実行

- sample-001.c の実行
  - 単に sample-001.c をコンパイルする (cc sample-001.c) と...
    - 「new_IList がない」などと怒られる
    - new_IList は ilist.c にある (分割されている..)
  - 単純な案 (同時コンパイル)
    - 一緒にコンパイルすると、自動的にまとめてリンクされる
    - cc sample-001.c ilist.c icell.c
  - 分割コンパイル (-c オプションをつけ .obj を作る)
    - cc -c ilist.c
    - cc -c icell.c
    - cc -c sample-001.c
    - cc sample-001.obj ilist.obj icell.obj
- ヘッダーファイルの役割
  - 型定義などを共有し、矛盾がおきないようにする

プログラムの説明(icell)

- □エラーの処理方針
  - ○エラーは報告だけはする
    - ▹途中で終了したりはしない(望ましい振舞)
  - ○エラー状態では、無難に「何もしない」
    - ▹報告もしない(本当は、報告した方がよいが..)
- □icell.h

  型 ICell の宣言 (int cell)
  - ▹次の要素をさす next と、整数データを保持する構造体
- □icell.c
  - ▹新規作成(new_ICell)
    - ◇領域の確保と初期化を行う
    - ◇確保できない場合は報告 (NULL を返す)
  - ▹開放(free_ICell)
    - ◇領域の開放を行う
    - ◇エラーチェックも行う(NULL かどうかの判定)
  - ▹出力(print_ICell)
    - ◇セルの内容を表示(中身に関しては、可能な限り中で対応する)

プログラムの説明(ilist)

□ ilist.h

型 IList の宣言 (int list)

- ▹ List の先頭の要素をさす top をもつ構造体
- ▹ top が NULL の場合は、空リストを意味する

□ ilist.c

- ▹ 新規作成(new_IList) : 空リストを作る
- ▹ 開放(free_IList) : リストを開放(中身も開放する)
- ▹ 出力(print_IList) : リストを出力、中身は ICell にまかせる
- ▹ 挿入(insert_IList) : リストの先頭に要素を追加
- ▹ 追加(append_IList) : リストの最後に要素を追加
- ▹ 検索(search_IList) : リストの中に要素があるかどうかを判定

C 言語でオセロ(2)

- rand
  - 疑似乱数を生成するライブラリ関数
    - 関数を呼び出す度に異なる値を答える
- playable (board, x, y, color)
  - 現状の board で, color が x, y にうてるかどうかを判定
- input1.c
  - できる所をデタラメに選んで、プレイする
-