ソフトウェア概論 A/B

-- 代入 --

数学科 栗野 俊一

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

- 前期の復習
  - プログラムとは
    - 命令を並べた物:命令の組み合わせによって様々な機能(複雑な事)が実現できる
  - プログラミングとは
    - プログラムを作成する事:計算機に自分の望みの振舞をさせられる
  - プログラミング言語とは
    - プログラムを記述する為の言語:計算機に対する命令の表現方法
  - C 言語とは
    - プログラミング言語の一つ:手続型/コンパイラ型:関数を記述
  - C 言語での命令とは(前期の内容では..後期では増える..)
    - 基本は関数呼び出しと、その制御構造(命令の実行順序を変更する指定)
    - 三つの制御構造がある(順接、条件分岐、再帰)
  - C 言語の関数とは
    - 命令の並びに名前を付けたもの:自分でも作れるし、予め用意されたものもある
  - C 言語でのプログラミング
    - 目的を実現するために関数を定義する事:最低限 main 関数が必要(Hello,World)

お知らせ

- 本日の予定
  - 代入
    - 代入命令
    - 変数宣言
    - 代入とメモリモデル
  - 代入による計算
    - プログラム設計パターン：入力・処理・出力
    - ステートマシンモデル (状態計算モデル)
- 本日の目標
  - 演習
    - 課題の提出

先週 (2011/09/24) の課題

- 先週 (2011/09/30) の課題
  - 課題 1:
    - ファイル名 : 20110930-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 成績処理を行う
    - 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)
    - 引数で自分の成績(S,A,B,C,D)をいれて、順位を表示させてみよ

今週 (2011/10/07) の課題

- 今週 (2011/10/07) の課題
  - 課題 1:
    - ファイル名 : 20111007-1-QQQQ.c (QQQQ は学生番号)
    - 内容 :

      二つの整数型の変数 m,n を宣言し、それにキーボードから値を入力する

      五つの整数型の変数 wa, sa, seki, sho, amari を宣言し、それに m と n の

      和、差、積、商、余りを計算した結果を代入する。

      変数 wa, sa, seki, sho, amari の結果を出力する
    - 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)

例題

- 例題：1 から n までの整数の和を求める関数を作る
  - int sum (int n) : 1 ～ n の和
- 考え方 (数学的帰納法を利用する)
  - 基本：素直な実装 (sample-001)
    - sum (1) の時を考える
    - sum (k - 1) の結果を利用して、sum (k) を求める
  - 蓄積型：頭からやる (sample-002)
    - とりあえず、0 を準備する
    - k - 1 まで加えた物に、残りの k ～ n までを加える
- 何が違うか？
  - 結果はとりあえず、同じになるようだが... ？
    - 動作を比較してみよう (sample-003, sample-004)
    - 関数の再帰呼び出しと、計算(add)の呼び出す順序が異なる
  - 基本型
    - 関数の再帰呼び出しが、「計算の順序を決め」、最後に計算を行う
    - 再帰呼び出し終って (n == 0 の時) も、やる事(足し算)が残っている
  - 蓄積型
    - 再帰呼び出しの順序が、計算の順序 (計算しながら動作する)
    - 再帰呼び出し終った時が計算が済んだ時 (引数 sum に値が入っている)

「計算」とは？

- 「計算」に対する二つの見方
  - 関数的観点 (sample-001 の「sum 関数の値」)
    - 計算は、式の「評価」をする事
    - 関数の値を計算する事は「評価」の基本的な振舞
    - 再帰呼び出しも、「関数の値の評価」に過ぎない
    - プログラミングは、値を計算する「関数の定義」を記述する
  - 手続的観点 (sample-002 の「sum 引数の値」)
    - 計算は、「値を変化」させる事
    - 再帰呼び出しは、「値を変化させる手順」を指定している
    - プログラムの実行順序と値の変化する順序が同じ
    - プログラミングは、「値の計算順序」を指定する
- (注意)「出力」は、「実行順」に行われる
  - 基本は、「蓄積型」に振る舞う (sample-005)
    - 実行順序が即ち、表示順序 (sample-006)
    - output_stars の呼び出し順序が表示順序

代入

- 代入とは？
  - 変数の値を「変更」する事
    - 言い替えると..？「変数は代入が行われると変数の値が変化する」
  - 変数は常に値を持つが..
    - 代入された前と後では「値が変化」する
- 変数に値を代入するには？
  - 代入文 (「変数名 = 式」) を使う (sample-007)
    - 式の値が計算された上で、変数に代入される
    - 「=(等号)」を使っているが、「等しい」という意味ではない
    - C 言語では「等しい」という意味には「==」を使う
  - 変数の値は何度でも参照可能 (sample-008)
    - 計算結果を何度も利用する時には変数に入れておく
  - 代入の右の式の中で自分自身の現在の値が利用できる (sample-009)
    - やっぱり、「等号」じゃない

代入と関数引数

- 関数引数は「代入」か？(sample-010)
  - 代入は、「変更」がおきる
    - 元々変数がもっていた値が「失われて」しまう
  - 関数引数は、変数の値を決めるが、「変更」するわけではない
    - 変数の値が保存されている (sample-011)
    - 同じ「名前」を利用していても「異なる変数 (アドレスが異なる)」

局所変数宣言

- 引数でない変数
  - ブロックの先頭で変数宣言すれば、新しい変数が利用できる
    - メモリの一部がその変数名に割り当てられる
  - 引数との違い
    - 宣言直後は値が定まっていない(何が入っているかは不明)
    - 必ず、値を代入してから利用すること(可能なら初期代入すること)
    - 「初期代入」'=,「変数宣言」+「代入」(実は微妙に違うが..)
- 名前の有効範囲
  - 「変数名」の有効範囲は宣言の後からブロックの終りまで
    - ブロックの外からはその名前が利用できない
    - 同じ名前で宣言しても異なるメモリになる(可能性が高い)
  - 名前とメモリは「独立」な事に注意
    - 名前は利用できなくてもメモリは利用できる(ポインターの利用)

入力・処理・出力

- 変数を利用したプログラムパターン
  - 入力・処理・出力の三つの部分からなる
    - 入力：変数を初期化する
    - 処理：(計算を利用しつつ..) 変数の値を変更する
    - 出力：変数の値を「結果」として出力する
  - 関数定義もこの形
    - 引数の値の設定、関数の本体、値の返却

ステートマシンモデル (状態計算モデル)

- 計算とは変数の値を変更する事である
  - ステート：変数の値の集合が、
  - 計算：ステートを変更する事
  - 計算の終了：ステートが望みの結果になっているという事
    - 求める結果が、どこかの変数の値になっていればよい
    - 画面への出力も「出力画面」という変数への「蓄積結果」と考える