ソフトウェア概論 A/B

-- オセロゲーム盤(2) --

数学科 栗野 俊一 / 渡辺 俊一

伝言

私語は慎むように !!

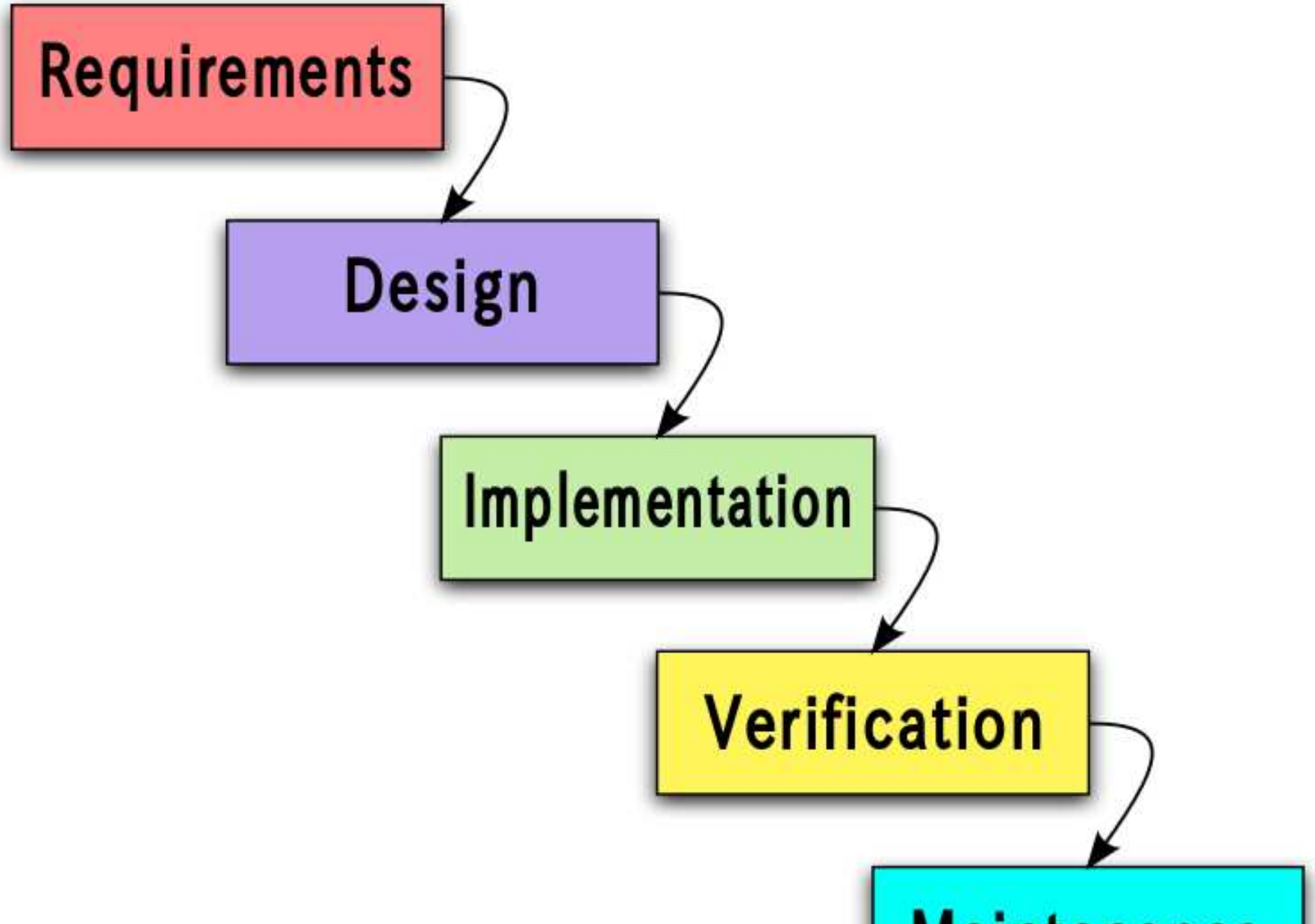
- 色々な「お知らせ」について
  - 栗野の Web Page に注意する事

    http://edu-gw2.math.cst.nihon-u.ac.jp/~kurino
- 講義開始前に済ませておく事
  - PC の電源を入れる
  - ネットワークに接続しておく事
  - 今日の資料に目を通しておく事
- 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
- 本日の CST Portal の出席パスワード : 20141107
  - 出席は成績に影響しませんが、折角の機能なので、使いましょう

前回 (2014/10/24) の復習

- 前回 (2014/10/24) の復習
  - オセロゲーム盤を作る (1)
    - プログラムの作成手順 : 要求分析/機能設計/コーディング/デバッグ/保守
  - オセロゲーム盤の設計
    - 8 x 8 の二次元配列での実装
    - 個々の桝には、白('o')/黒('x')/空白(' ')の何れかが入る
    - ゲーム盤の初期化 / ゲーム盤の表示
    - リファクタリング

ウォータフォールモデル(Waterfall Model)



ソフトウェア開発手法

- □ウォータフォールモデルとは (What)
  - ○ソフトウェア開発のモデル(ソフトウェアの開発手順の流れ)の一つ
    - ▹最も古典的(ようするに古い!!) : 現在は利用されていない
    - ▹でも、「基本概念」は重要 : 新しいモデルはこれの改良として実現される
- □ウォータフォールモデルの概要(How to)
  - ○ソフトウェアの開発ステップを次の 5 つのステップの分けて考える
    - ▹要求分析 : プログラムの目的や利用者の望みを確認する → (外部)仕様書
    - ▹機能設計 : 目的を実現するための手段やデータ構造、アルゴリズムを考える → 内部設計書
    - ▹コーディング : データやアルゴリズムをプログラミング言語で表現する → プログラム
    - ▹デバッグ : プログラムが要求仕様を満すかを確認 → 検査表
    - ▹保守 : プログラムを利用者に提供できるようにする → マニュアル
  - ○個々のステップは、滝(Waterfall)の水が流れるように上流から下流へ流れる
    - ▹「後戻り」してはいけない !! → その分、開発が遅れるから (Why)
    - ▹実際は、「後戻り」は回避不能 →「後戻り」を考慮したモデルへ改良

プロトタイピング

- □ プロトタイピングとは(What)
  - ○ 将来完成する予定のソフトウェアの不完全なモデル(プロトタイプ)を作成すること
    - ▹ 仕様が完全でなくても、その仕様で、一度プログラム(プロトタイプ)を作ってしまう(一旦、滝を下ってしまう)
- □ プロトタイピングの利点(Why)
  - ○ 上流(要求分析や設計の段階)で分かり難い事が明確になる
    - ▹ プログラムのイメージ、問題点、難易度..
  - ○ 要求分析は難しい
    - ▹ 利用者(顧客)の希望を明確にして仕様の形にまとめるのは難しい
    - ▹ 利用者は(当然、プログラムについて..)素人なので、頓珍漢な事を言う
    - ▹ それを(同じ位出来の悪い..)営業がまとめて来た日には.. (8_8)
  - ○ 性能の確認も難しい
    - ▹ 「実行してみないと解らない」事は、残念ながら実際にある
- □ プロトタイピングの利用(How to)
  - ○ プロトタイプを利用者に見せて、「イメージ」を確認させる → 仕様の詳細を確定
  - ○ プロトタイプの性能から完全版の性能を図る → 必要ならアルゴリズムの改良
  - ○ 製品版の初版として扱う(プロトタイプを改良して製品版を作る)
    - ▹ hello.c 方式 !! → プログラム作成の無駄が軽減できる(アマ向け)
    - ▹ 余りすすめられない → プロトタイプは「やっつけ」の事が多いので信頼できない(プロは作り直しが吉)

テストファーストとリファイクタリング

- テストファースト(ドリブン)プログラミングとは(What)
  - プログラムを作る前にテストを作り、テストをしながらプログラム完成させる手法
    - 空っぽのプログラム(hello.c)で、テストを動かす(必ず失敗する!!)
    - プログラムを改良して、再度テスト(成功すれば完成 !!)
  - 「プログラムの完成」でなく「テストの全成功」が目的になる(気持良い)
    - 目標がたてやすく、進行状況も分かり易い
  - テストが先なので、大胆な改良が可能(Why)
- リファイクタリングとは(What)
  - プログラムの機能を変更せずに、内部の構造を改良する事
    - → これによって、プログラムの改良がしやすくなる(Why)
  - どうせ改良するなら、機能もふやせば？
    - → 機能をふやすとバグ(プログラムの誤り)が増えるかもしれない
  - リファイクタリングの利点
    - 「機能追加」と「構造の整理」を分割する事により、問題を減らす
  - テストファースト手法と組で利用される事が多い
    - テストコードが再利用できる(機能が増えてないから/生産性の向上)

アジャイルソフトウェア開発

- アジャイルソフトウェア開発とは(What)
  - 迅速かつ適応的にソフトウェア開発を行う軽量な開発手法群の総称
    - 色々なテクニックを組合せている
    - 例：テストファースト/プロトタイピング/リファクタリング..
- 文献「アジャイルソフトウェア開発の奥義」
  - ロバート・C・マーチン著 / 瀬谷 啓訳
  - <雑談>
    - 瀬谷さんとお話した事がある(サイン本をもっている)
    - 山中君(栗野研卒、君等の先輩)が、この本のプログラム作成で協力
- Othello もアジャイルで :-)

お知らせ

- 本日の予定
  - オセロゲーム盤を作る(2)
  - 配列と関数引数
- 本日の目標
  - 演習
    - オセロゲーム盤を動かしてみる

前回 (2014/10/24) の課題

- 前回 (2014/10/24) の課題
  - ありません

本日の課題 (2014/11/07)

- 本日 (2014/11/07) の課題
  - 課題 1:
    - ファイル名 : 20141107-1-QQQQ.c (QQQQ は学生番号)
    - 内容 : 本日(2014/11/07) の段階でまでに作られた othello.c (v?.c)
    - 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)

関数引数(復習 1)

- □ 単純型変数と関数引数：「副作用」がない
  - ○ 関数呼び出し時に、実引数を渡す事ができる
    - ▷ 実引数には「式」が指定できる
    - ▷ 実引数の「式」の「値」が計算され、「値」が渡される
  - ○ 関数定義の仮引数変数は、実引数の「値」のコピーを持つ
    - ▷ 仮引数の値を変更する事はできる(関数内で仮引数の値は変る)
    - ▷ この値の変更は、呼出し元には影響しない(副作用がない)
- □ 注意(重要なので繰返し)
  - ○ 関数呼出し時に実引数として単純変数を指定しても..
    - ▷ 「変数そのもの」が関数に「渡たされる」わけでは「ない!!」
    - ▷ 「渡される」のは「変数の値」でしかない

関数引数(復習 2)

- 構造体型変数と関数引数：「副作用」がない
  - 関数の実引数に、構造体型の変数も指定できる
    - 単純型変数と同様、値が渡されるだけ
- 配列と関数引数：「副作用」が起きることがある
  - 関数の実引数に、配列名を指定できる
    - 注意：「配列型変数」というものはない
    - 関数内で、「配列要素の値を変更」すると、呼出し元の「配列要素の値」も変化する
  - 副作用：関数呼出しによって、呼出し元の変数の値が変化
    - 危険：プログラムを読む時に(値は変化したか?に)「気を付ける」必要がある
    - 便利：代入以外で、呼出し元の変数を変更する手段になっている (cf. scanf)
    - 効率：コピーをしないので速い(これのために利用される事が多い[必要悪])
- 「配列の引数渡し」の振舞
  - <<取り敢えず>> 配列名を渡すと、「配列が丸々渡されるようにみえる」と理解
    - 関数内で、「要素の値が取り出せるだけでなく、変更も可能」(他は変更不可)