- FreeMind Programming
  - (L) 20090106/7/9/12/0528/31/111013 zioi(a)ieee
  - 概要：FreeMindを使ってプログラムを作る方法の整理
  - 効果
    - プログラムの構造が見やすくなる
      - 2次元的に、紙・画面を有効に使う
      - テキストエディタなどでは、構造はプログラマの頭の中にある
    - 編集が視覚的になる　移動が視覚的
    - プログラムブロックに名前を付けることができる（コメントのみのノード）
    - {や;のなど、構造から想定できるものを省略し、自動挿入することにより、それらの入力ミス・エラーが少なくできる
  - Perl
    - mm2pl.mm：変換プログラム
    - mm2pl解説.mm
    - テスト用プログラム：mm2pl.mm自体
    - 課題
      - 網羅的テスト用プログラムを作る
      - pl2mmの逆変換を作る(Perlの書き方が多彩で、あきらめるのが賢明)
  - C
    - mm2c.mm：変換プログラム
      - 090108：mmデータ、single open tag, single close tag, self-closing open tagのいずれかとする。つまり、<node..>data</node>というデータは無いと仮定する。
      - 090107：一度tag level + dataのファイルに変換すると処理を明確に分割できる。mm-file -> mm2lt -> lt2c -> C
    - mm2c解説.mm
    - c-sample?.mm：テスト用プログラム
      - c-sample1.c：switchのテストも入れときたい
      - c-sample2.mm：#ADD_SEMICOLONのテスト用
    - 課題
      - header fileを使わなくてよいようにできるとよい（情報分散を避けるため）
      - c2mmの逆変換を作る
        - 1 FreeMindにノードデータ入力方法が用意されているとよい
        - 2 FreeMindへの手入力エミュレータを作る
        - 3 FreeMindのデータを作る（一番遠回り、FMを詳細に知らないとトラブルの可能性あり
  - 他の言語
    - mm2cをC++、Javaでも使えるか試す
    - Scalaではコメントノードの下に｛｝ブロックを作れないので、mm2plを使う
    - Makefileも
  - 利用のヒント
    - FreeMindで全部表示しても充分見渡せる程度のプログラムサイズにするのが賢明である
    - プログラムすべてをひとつのmmファイルにする方向も意味ある
  - 課題：
    - コンパイル時エラーなどは変換後のファイルに対するものなので、FreeMindではエラー部分の判断が少し不便になる。エラーの行番号と内容を見ただけで、どこかわかる程度のプログラムサイズにするのがよい。
    - FreeMindでは、"などの記号が"&..;"で内部表現されているが、分かっている分だけしか元の文字に戻していない
    - 日本語でコメントを書けるが、変換後ファイルでは"&#x....;"でunicode表記のままになる
    - 階層構造データを元に構文チェックを行い色表示するとよいのでは（090529秋葉）
  - 他の同様な方法
    - UML
      - UMLでのプログラム挿入は視覚的だが、ブロックレベルで粒度が違うだろう
      - UMLプログラムでは、オブジェクト間の関係を線でつなぐのが重要な作業（FMPではプログラム中に書く）
    - Visual Studioでは{..}の閉じる、開く、アウトライン表示、移動ができきる（090529秋葉）
  - 参考
    - .mmのファイル形式は他のmind mapと共通では？（090529秋葉）