Laboratorio 2
(Sintassi Astratta in C e Introduzione a OCaml)

Sommario: 23 Marzo, 2018

- Overview di OCaml: Parte 1
- Alberi di Sintassi Astratta in C: Dcl

Overview di OCaml: Top Level Interpretation Cycle

- Eseguiamo ocaml in Unix ed entriamo in Top Level Interpretation Cycle
- Ambiente Interattivo: un blocco virtuale, Scope Statico, un ciclo
  - input: Termine (i.e. Espressione, Programma) T terminato con ';;'
  - Valutazione di T;
    - T è analizzato dal FrontEnd
    - T è Compilato in Bytecode OCaml T_B
    - T_B è Interpretato e valutato ad un valore V;
  - output: *nome*: *tipo* = V;
- Il ciclo input-valutazione-output si ripete:
  - Fintantochè T è OCaml-corretto e valutabile ad un valore;
  - Valutazioni non terminanti per termini divergenti;
  - control-c per interrompere interprete e tornare in input (all'inizio del ciclo)
  - exit(n);; – con n intero per chiudere il ciclo ed uscire dall'Ambiente

Overview di OCaml: Parte 1

- OCaml è un Linguaggio Multiparadigma:
  - sottoLinguaggio Funzionale (puro): Calcolo Funzionale
  - sottoLinguaggio Imperativo: stato, modificabili e sequenza;
  - sottoLinguaggio Object Oriented: Classi, Oggetti ed Ereditarietà;
  - integrati in accordo a specifiche regole di composizione dei costrutti
- Nel Laboratorio useremo solo il sottoLinguaggio Funzionale
- Oggi vediamo (vedi Listing OCaml1)
  - Operatori primitivi e binari, infissi usabili anche in forma prefissa.
  - Naming per valori costanti: Costrutto "let ide = T".
  - Valori Primitivi Atomici per tipi: int, float, bool, char.
  - Linguaggio fortemente tirato: ogni termine ha uno e un solo tipo.
  - Il tipo determina operazioni applicabili ed uso del valore.
  - Valori Primitivi Strutturati per tipi: string, tuple e liste.
  - Cautela nella sintassi concreta.
  - Valori Funzione: Astrazioni fun e Naming.
  - Funzioni Monadiche e Currying.
  - Ricorsione e operatore rec.s
  - Blocchi: Costrutto "let ide = T1 ... ide = Tn in T".

Alberi di Sintassi Astratta in C

- Esaminiamo il Listing Dcl.h incluso in ciascuno dei folder Bad e Good di Dcl.zip,
- Dcl.h contiene (in entrambi i casi) :
  - Definizioni dei tipi per gli Abstract Tree (AT) delle dichiarazioni di SmallC
  - Definizioni dei tipi per implementare gli AT, sopra.
    - Bad: ParseTree (sconsigliata perchè ...)
    - Good: Struct ad hoc per ogni tipo (raccomandata perchè ...)
  - Intestazioni (i.e. signature) delle operazioni sugli Abstract Tree da implementare
- Stesura del file Dcl.c con il codice delle operazioni per il Dcl.h raccomandato (quello in Good)
  - Tempo rimastoci:20'
  - Bilancio: 18 partecipanti
    - Partecipanti che terminano la soluzione Good: 1
    - Partecipanti che completano 50% di Good: 3
    - Partecipanti che completano 50% di Bad: 3
    - I rimanenti svolgono parti e non conoscono il costrutto union del C
- Completare la stesura della propria soluzione (studiandosi il c. union del C)
- Completare Esercizio 2 di Laboratorio1: scrivere il main C richiesto.