Università degli Studi di Udine
Department of Scienze Matematiche, Informatiche and Fisiche
Anno Accademico 2025/2026

Linguaggi e Compilatori - Parte 1

Gruppo 25 - 3

Jacopo Danielis, 162553 danielis.jacopo@spes.uniud.it
Aleksa Aleksic, 161711 aleksic.aleksa@spes.uniud.it
Ludovico Gerardi, 162367 gerardi.ludovico@spes.uniud.it

1 Descrizione della soluzione

1.1 Specifiche

Per il progetto sono stati utilizzati il linguaggio C11, BNFC 2.9.6.1, Bison 3.8.2 e Flex 2.6.4.

1.2 Implementazioni notevoli

- Il Mantenimento delle infomazioni sulle sezioni e sui campi avviene attraverso una lista concatenata, rappresentante le sezioni, di liste concatenate, a loro volta rappresentanti i campi.
  Non tutti i campi associano direttamente il nome al valore, in quanto alcuni possono mantenere un riferimento ad un altro campo, derivante dall'uso di variabili, locali e non locali, oppure da direttive di `inherit`.
  Il parser non costruisce un AST, bensì agisce sulla variabile che mantiene la lista delle sezioni, costruendo queste ultime man mano che il parsing prosegue.
  L'unica eccezione avviene durante il riconoscimento dei valori base oppure della notazione "$", per i quali viene prodotto un nodo contenente il valore, che viene poi utilizzato nella creazione del campo corrispondente.

- Per il riconoscimento di import circolari viene utilizzato `sys/stat` in modo da riconoscere l'uguaglianza tra due file ed evitare di importare lo stesso file due volte.
  Questa strategia non porta a una perdita di informazione e previene i cicli, poichè importare nuovamente lo stesso file non avrebbe effetti sui legami campo-valore.
  Siccome le funzioni utilizzate sono proprie di sistemi Unix e non di Windows, sono state rese disponibili compilando con l'opzione
  `-D DETECT_IMPORT_CYCLES=1`.

- Inoltre, viene implementata la funzione `saveToFile`, la quale salva l'output del pretty-printer in un nuovo file, su cui eseguire il round-trip-test, ovvero poter controllare che il file iniziale parsato e la versione ottenuta da `saveToFile`, a sua volta parsata, producano lo stesso output.

1.3 Pretty-Printer

L'implementazione del Pretty-Printer richiesto avviene attraverso la modifica dei file `Printer.c` e `Printer.h` con l'aggiunta della funzione `printFromBindings`, in modo tale da andare a stampare il codice a partire dalla struttura dati includendo i commenti correttamente.

All'interno della struttura dati, la lista contentente i binding tra commenti e testo è costruita tramite coda LIFO, mentre la stampa deve avvenire in maniera FIFO. Conseguentemente, il codice di print inverte l'ordine degli elementi nella lista in modo tale da poter stampare le sezioni e i field correttamente. L'idea di base è quella di stampare gli elementi, andando inoltre a:

- etichettare ogni commento stampato per evitare ripetizioni;
- stampare tutti i commenti non etichettati prima dell'inizio di una sezione, in modo da includere quelli all'inizio del file;
- stampare i commenti inline presenti alla fine dell'apertura di una sezione o di un field e alla fine della chiusura delle sezioni e dei field;
- stampare tutti gli eventuali commenti presenti dopo la chiusura dell'ultima section, ovvero eventuali commenti alla fine del file.

Una volta stampati i commenti, la struttura dati dei binding viene di nuovo invertita, in maniera da ripristinare l'ordine iniziale dei valori salvati all'interno, per continuare con la stampa.

Il salvataggio dei commenti da parte del lexer avviene in una struttura dati apposita.
La stampa nella posizione corretta è garantita da un attributo `line`, in tal modo il printer è capace di ricostruire il corretto ordinamento dei commenti all'interno di sezioni e field.