Politecnico di Bari

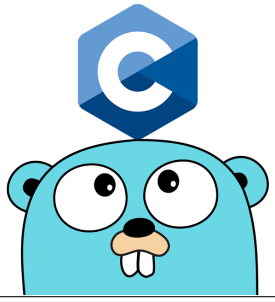
DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE
Corso di Laurea Magistrale in Ingegneria Informatica

Formal Languages and Compilers

C2Go:
Guida all'uso

Docenti:
Prof. Floriano Scioscia

Componenti:
Salvatore Bufi
Angela Di Fazio



Anno Accademico 2020–2021

Indice

Capitolo 1

Compilazione

Prima di poter utilizzare il transpiler *C2Go* è necessario compilare i codici sorgenti. Questa operazione può essere effettuata in maniera manuale mediante l'utilizzo del Makefile.

1.1 Compilazione con l'uso del Makefile

Per effettuare la compilazione bisogna effettuare i seguenti step:

1. Accedere da terminale alla directory `c-to-go`.
2. Eseguire il comando `make` che permetterà di compilare automaticamente i sorgenti.

1.2 Compilazione manuale

Per effettuare la compilazione manuale bisogna effettuare i seguenti step:

1. Accedere da terminale alla directory `c-to-go`.
2. Generare il parser e il relativo header con il comando `bison -defines=token.h -o parser.c parser.y`.
3. Generare lo scanner con il comando `flex -o scanner.c scanner.l`.
4. Compilare il transpiler con il comando `gcc parser.c scanner.c tree.c symtab.c semantic.c translate.c -lfl -o compiler`.

Capitolo 2

Utilizzo

Una volta effettuata la compilazione del transpiler il programma può essere eseguito mediante il seguente comando:

```
./compiler path_input_file
```

Ad esempio:

```
./compiler test/programs/bubble_sort.c
```

Dopo l'esecuzione, in assenza di errori, sarà generato il programma target nel file `traduzione.go`.

2.1 Guida all'utilizzo

E' possibile visualizzare una guida all'utilizzo del compilatore mediante il flag `-help` (o `-h` analogamente).

```
./compiler --help
```

Verrà così visualizzata la seguente guida all'uso:

```
Usage: ./compiler [options] file
options:
 --help                  Diplay this information.
-h                       Diplay this information.
-s                       Print Symbol Table.
-t                       Print Abstract Syntax Tree.
```

Tale guida presenta la sintassi del comando e gli altri flag disponibili.

2.2 Flag

I flag previsti dal compilatore sono stati utilizzati durante la fase di debug per verificarne il corretto funzionamento.

- `-s`: questo flag permette di mostrare sullo standard output il contenuto della symbol table alla chiusura di ogni scope. Esempio:

```
SYMBOL TABLE     scope: 1
--------------------------
simbolo: return          tipo: int
simbolo: b       tipo: float
--------------------------
SYMBOL TABLE     scope: 0
--------------------------
simbolo: f       tipo: int
simbolo: main    tipo: void
--------------------------
```

- `-t`: questo flag permette di mostrare sullo standard output una rappresentazione del codice sorgente ottenuta attraversando l'abstract syntax tree. Esempio:

```
int f() {
        float b=1.2;
        printf("%f", b);
        return 1;
}

void main() {
        int a=10;
        printf("%d", a);
        f();
}
```

Nota: Questa guida è pensata per l'utilizzo del transpiler in ambiente Linux.