Progetto per il Laboratorio di Programmazione
Un interprete per un piccolo C

Modulo IV: L'esecutore (EX)

Stefano Guerrini

A.A. 2003/04 – Canale P-Z
Versione del 31 agosto 2004

1 Modulo IV: L'esecutore (EX)

L'esecutore si compone di una funzione principale

```
void exec_program (void);
```

che esegue un programma CCINO precedentemente letto e trasformato dall'analizzatore sintattico.

1.1 Regole di esecuzione dei comandi CCINO

Le regole da seguire per l'esecuzione di un programm CCINO sono le stesse del linguaggio C, con l'aggiunta delle regole per l'eseuzione dei comandi **read**, **write**, **wsp** e **wnl** di CCINO. Si ricorda che, il comando

- **read**(x) legge un valore da input e lo memorizza nella variabile x;
- **write**(x) stampa il valore dell'espressione e;
- **wsp**(x) stampa un numero di spazi bianchi pari al valore di e;
- **wnl**(x) stampa un numero di new-line pari al valore di e.

1.2 Compilazione di un programma CCINO

Un programma CCINO è compilabile mediante un compilatore C, a patto di includere nel programma l'header file `ccino-cc.h` riportato qui di seguito

```
/* -*- Mode: C -*- */
/* Time-stamp:  <ccino-cc.h 04/08/12 14:17:59 guerrini@guerrini.dsi.uniroma1.it> */

/************************************************************
 * Progetto di Laboratorio di Programmazione
 * Stefano Guerrini - AA 2003-04
 *
 * Header per la compilazione di un programma scritto in ccino
 ***********************************************************/
```

```
#ifndef __CCINO_CC__
#define __CCINO_CC__

#include <stdio.h>

/*
 * I comandi read e write sono definiti come macro
 */

#define read(n)  scanf("%d", &n)
#define write(e) printf("%d", e)

/*
 * I comandi wrsp e wrnl sono definiti come funzioni
 */

void wsp (int e);
void wnl (int e);

#endif
```

e di compilare il programma CCINO insieme al file `ccino-cc.c` riportato qui di seguito

```
/* -*- Mode: C -*- */
/* Time-stamp:  <ccino-cc.c 04/08/12 14:17:50 guerrini@guerrini.dsi.uniroma1.it> */

/*************************************************************
 * Progetto di Laboratorio di Programmazione
 * Stefano Guerrini - AA 2003-04
 *
 * Libreria per la compilazione di programmi scritti in ccino
 *************************************************************/

#include <stdio.h>
#include "ccino-cc.h"

/*
 * Le funzioni wrsp e wrnl sono definite come funzioni
 */

void wsp(int e) {
    while (e--) {
        putchar(' ');
    }
}

void wnl(int e) {
    while (e--) {
        putchar('\n');
    }
}
```

Si osservi che le funzioni **read** e **write** sono definite mediante due macro, mentre la **wsp** e la **wnl** sono implementate da due funzioni.

Usando il compilatore gcc, il comando

```
gcc -g -x c -include ccino-cc.h ccino-cc.c prova.ccino -o prova
```

produce l'eseguibile `prova` corrispondente al programma CCINO contenuto nel file di nome `prova.ccino` (a patto che i file `ccino-cc.h` e `ccino-cc.c` si trovino nella directory in cui si sta comppilando `prova`).

1.3 Correttezza dell'esecutore

Il test chiave per verificare la correttezza dell'esecutore implementato è verificare che l'ouput dell'esecuzione dei programmi di esempio contenuti nella corrispondente appendice e pubblicati sul sito web del laboratorio sia uguale a quello prodotto dai corrispondenti programmi generati dagli esempi quando compilati seguendo le istruzioni riportate in sezione 1.2.

1.4 L'ambiente di valutazione di una funzione

Uno dei punti chiave da capire per implementare l'esecutore e' che cosa è e come implementare l'ambiente di valutazione di una funzione. L'ambiente di valutazione di una funzione è la corrsipondenza tra le variabili che occorrono all'interno della funzione e le locazioni di memoria in cui leggere e scrivere i corrispondenti valori.

Nell'ambiente di valutazione possiamo quindi distinguere due parti:

- l'ambiente globale, per le variabili globali del programma;
- l'ambiente locale, per i parametri e le variabili locali della funzione.

Tenendo conto di come è stato indicato di rappresentare le variabili nelle specifiche dell'analizzatore sintattico, sia l'ambiente globale che quelli locali possono essere rappresentati con dei vettori di interi (in CCINO gli interi sono l'unico tipo di dato) tali che la locazione di memoria della i-esima variabile è li-esimo elemento del vettore. Si osservi però che, mentre l'ambiente globale può (deve) essere allocato dall'esecutore una volta per tutte prima di cominciare l'esecuzione dei comandi del main, l'ambiente locale di una funzione può (deve) essere allocato solo al momento della chiamata della funzione. Inoltre, alla fine delle valutazione della funzione, oltre a garantire che il controllo ritorni all'istruzione che segue la chiamata della funzione, occorre ripristinare l'ambiente locale valido al momento della chiamata.

Una maniera per garantire che l'ambiente locale sia sempre quello corretto è quella di organizzare gli ambienti locali in uno stack la cui testa contiene sempre l'ambiente locale della funzione che si sta eseguendo. Questa soluzione corrisponde a quella usata da un esecutore iterativo del programma e più in generale è l'unica adottabile se anzichè eseguire direttamente il codice si sta generando codice per un linguaggio a basso livello (ad esempio, codice assembler o direttamente codice macchina). Se l'esecutore viene però realizzato per mezzo di una serie di funzioni ricorsive, per rappresentare gli ambienti locali si può anche sfruttare lo stack delle chiamate delle funzioni del C, facendo attenzione che la funzione che si occupa di valutare le funzioni del programma CCINO allochi correttamente l'ambiente locale al momento della chiamata di una funzione e lo deallochi alla fine.

1.5 Verifica

Per verificare l'esecutore si utilizzerà il main riportato in Appendice A.

Se si sta lavorando su di un sistema linux con compilatore gcc, supponendo che il nome del file che contiene il codice del valutatore di espressioni è `exec.c` e che i precedenti moduli sono contenuti nei file `tavole.c`, `lexan.c`, `syntan.c`, il comando per la compilazione del programma di test è:

```
gcc -g ccino-IV.c ccinodefs.c tavole.c lexan.c syntan.c exec.c -o ccino-IV
```

Come risultato della compilazione si otterrà l'eseguibile `ccino-IV` che legge ed esegue un programma CCINO.

Si osservi che per verificare questo modulo servono tutti i moduli precedenti. Se tali moduli non sono stati implementati, o sono stati inviati moduli non funzionanti, si utilizzeranno i corrispondenti moduli sviluppati dal docente.

A Modulo IV: Verifica dell'esecutore di programmi CCINO (EX)

Il main che verrà utilizzato per verifcare l'esecutore è riportato alla fine di questa appendice. La compilazione di questo main insieme a tutti i moduli del progetto genera un eseguibile che riceve come argomento sulla linea di comando il nome del file con il programma CCINO da eseguire.

Per garantire che il modulo dell'analizzatore lessicale continui a funzionare (si ricorda che tale modulo supponeva che il programma da analizzare fosse fornito sullo standard input), il main assegna lo standard input al file con il programma CCINO durante la fase di analisi lessicale e sintattica e, prima di passare alla fase di esecuzione, riassegna lo stdin al suo valore di default—ovvero, a meno di reindirazzamenti sulla linea di comando, alla console[1]. Le precedenti operazioni richiedono la chiamata di alcune funzioni di basso livello per la manipolazione di file, in particolare le funzioni `dup` e `dup2`. Senza entrare troppo in dettaglio sul funzionamento di queste funzioni ricordiamo che, a basso livello, ad ogni file è associato un descrittore univocamente determinato da un indice numerico (la posizione del descrittore nella corrispondente tavola). Nel main riportato nel seguito, la prima chiamata di `dup` crea una copia del descrittore di default di stdin (l'indice del descrittore di stdin è ottenuto mediante la funzione `fileno` al momento della definizione di `fd_stdin`); quindi, la `freopen` apre il file del programma assegnandolo allo stdin; infine, la chiamata a `dup2` ricopia nella sua posizione originale il descrittore di default di stdin inizialmente salvato nella posizione `fd_temp`, reindirazzando così stdin al suo default.

Le funzioni `dup`, `dup2` e `fileno` eseguono delle chiamate di sistema, per questo motivo il codice qui riportato può non funzionare su tutti i sistemi operativi e con tutti i compilatori. La versione riportata funziona sotto Linux con gcc (ma dovrebbe funzionare anche con un qualsiasi altro compilatore per Linux). Per compilare sotto Windows con Visual C, è necessario sostituire

```
#include <unistd.h>
```

con

```
#include <io.h>
```

ad esempio, commentando la linea della prima include e decommentando la linea della seconda nel sorgente che segue.

```
/* -*- Mode: C -*- */
/* Time-stamp:  <ccino-IV.c 04/08/31 16:50:27 guerrini@guerrini.dsi.uniroma1.it> */

/************************************************************
* Progetto di Laboratorio di Programmazione
 * Stefano Guerrini - AA 2003-04
 *
 * Main per la verifica del modulo IV: l'esecutore (EX)
 * Per la compilazione servono i seguenti file:
 *   ccinodefs.h       header principali defs
 *   ccinodefs.c       alcune var globali e funzioni di utilita
 *   ccino-IV.c        questo file
 *   tavole.c          tavole dei simboli - modulo I (TS)
 *   lexan.c           analizzatore lessicale - modulo II (AL)
 *   syntan.c          analizzatore sintattico - modulo III (AS)
 *   exec.c            esecutore - modulo IV (EX)
 * Per la compilazione su linux con gcc
 *   gcc -g ccino-IV.c ccinodefs.c tavole.c lexan.c syntan.c exec.c -o ccino-IV
 * crea l'eseguibile ccino-IV con le info necessarie per il debugging
 ************************************************************/
```

[1]Si osservi la differenza con il main del modulo dell'analizzatore sintattico dove, dato che la stampa non richiede operazioni di lettura, non è necessario riassegnare lo stdin al suo valore di default.

```
#include <stdio.h>
#include <stdlib.h>

/* header file per unix/linux */
#include <unistd.h>
/* header file per windows visual C e compilatori compatibili */
/* #include <io.h> */

#include "ccinodefs.h"

int main(int argc, char *argv[]) {
    /*
     * Esegue un programma CCINO letto da un file
     * passato come argome sulla linea di comando
     * L'eseguibile generato deve essere chiamato nel seguente modo
     *    ccino-IV fin
     * dove fin e' un file che contiene il programma CCINO
     * Il reindirizzamento di stdin e' ottenuto mediante le funzioni
     * di basso livello dup e dup2 e la freopen
     * Queste funzioni possono dipendere dal sistema.
     * Vedere i commenti delle include per compilare su Linux o
     * su Windows con Visual C.
     */

    int fd_stdin = fileno(stdin), fd_temp;
    struct descr_fun *fund_main;

    --argc; ++argv;
    if (argc == 0) {
        fprintf(stderr, "Specificare il nome del file da eseguire");
        exit(-1);
    }

    /*
     * Associa il file del programma al descrittore di stdin
     * salvando il descrittore di default di stdin in fd_temp
     */

    /* duplica il descrittore del file stdin di default */
    if ((fd_temp = dup(fd_stdin)) == -1) {
        perror("dup stdin");
        exit(-1);
    }
    /* riassocia stdin al file da eseguire */
    if (freopen(*argv, "r", stdin) == NULL) {
        fprintf(stderr,
                "Impossibile aprire il file di input \"%s\"\n",
                *argv);
        exit(-1);
    }

    /*
     * esegue il parsing leggendo dal file assegnato allo stdin
     */

    fund_main = parse_program();
```

```
    /*
     * ripristina lo stdin di default
     */

    /* chiude il descrittore del programma */
    if (close(fd_stdin) == -1) {
        perror("close stdin rediretto");
        exit(-1);
    }
    /* ripristina il descrittore di stdin salvato if fd_temp */
    if (dup2(fd_temp, fd_stdin) == -1) {
        perror("dup2 stdin salvato");
        exit(-1);
    }

    /*
     * esegue il programma
     */
    exec_program();
}
```