Progetto per il Laboratorio di Programmazione
Un interprete per il linguaggio PINO

Modulo I: Le tavole dei simboli (TS)

Stefano Guerrini

A.A. 2002/03 – Canale P-Z
Versione del 20 giugno 2003

1 Modulo I: Le tavole dei simboli (TS)

Durante la lettura di un programma occorre individuare se un dato identificatore corrisponde ad un nome di variabile globale, ad un nome di funzione, ad un nome di variabile locale, o ad una parola chiave e recuperare alcune informazioni associate a tale variabile o funzione (ad esempio, il numero di argomenti della funzione). Le strutture dati che permettono di gestire queste informazioni sono le *tavole dei simboli*.

Nell'implementazione di PINO noi utilizzeremo due tipi di tavole dei simboli: quelle implementate mediante vettori e quelle implementate mediante strutture dinamiche. Dove e come utilizzare tali tavole sarà chiarito nella specifica dei successivi moduli del progetto.

1.1 Tavole dei simboli implementate mediante vettori

La prima tavola di simboli o chiavi da implementare è realizzata con un vettore di stringhe—ogni chiave è una stringa. Il file `pinodefs.h` contiene le seguente definizione per una struttura che implementa una tavola con al massimo `MAX_KEYS` (anche questa costante è definita in `pinodefs.h`).

```
/* Struttura per la rappresentazione di una tavola mediante un vettore */
struct tavola {
    unsigned n; // elementi nella tavola
    char *keys[MAX_KEYS]; // vettore con le stringhe nella tavola
};
```

Dove `n` è il numero di elementi attualmente contenuti nella tavola ed il vettore `keys` contiene nelle sue prime `n` posizioni le stringhe delle chiavi memorizzate nella tavola.

Si devono scrivere le seguenti procedure:

```
void init_tavola(struct tavola *ptv);
/* inizializza la tavola *ptv */

unsigned avail_tavola(struct tavola *ptv);
/* numero di spazi ancora disponibili nella tavola *ptv */

int ins_key_tavola(struct tavola *ptv, char *key);
/* aggiunge la chiave key alla tavola se non gia' presente
 * ritorna l'indice della posizione della chiave inserita
```

```
 * o -1 nel caso in cui key e' gia' presente nella tavola o
 * non c'e' posto per inserirla
 */

int search_key_tavola(struct tavola *ptv, char *key);
/* cerca la chiave key nella tavola *ptv
 * ritorna la posizione della chiave nella tavola
 * o -1 se la chiave non \'e presente
 */
```

Il cui comportamento è descritto nei corrispondenti commenti.

1.2 Tavole dei simboli implementate mediante liste

Le tavole delle variabili e delle funzioni definite dall'utente verranno implementate mediante liste. Le strutture dati che servono per definire tali liste sono riportate qui di seguito (anche queste definizioni sono contenute in `pinodefs.h`).

```
/* Descrittore di funzione */
struct descr_fun {
    char id[MAX_ID_LEN]; // il nome della funzione
    struct tavola tv; // tavola con il nome degli argomenti della funzione
    struct expr *exp; // l'espressione associata alla funzione
};

/* Nodo lista descrittori di funzioni */
struct nodo_lista_dfun {
    struct descr_fun *pdf;
    struct nodo_lista_dfun *next;
};

/* Lista descrittori di funzioni */
typedef struct nodo_lista_dfun *lista_dfun;

/* Descrittore di variabile */
struct descr_var { // descrittore di variabile globale
    char id[MAX_ID_LEN]; // il nome della var
    struct expr *exp; // l'espressione associata alla var
};

/* Nodo lista descrittori di variabile */
struct nodo_lista_dvar {
    struct descr_var *pdv;
    struct nodo_lista_dvar *next;
};

/* Lista descrittori di variabile */
typedef struct nodo_lista_dvar *lista_dvar;
```

Ogni funzione ed ogni variabile ha un descrittore di tipo `desr_fun`, se funzione, o `descr_var`, se variabile. Il dettaglio dei dati contenuti in questi descrittori verrà analizzato in seguito (nella parte di analisi sintattica). Per il momento è sufficiente osservare che ciascun descrittore contiene un vettore di caratteri di dimensione `MAX_ID_LEN` (anche questa costante è definita in `pinodefs.h`). In tale vettore va memorizzato il nome della variabile o funzione rappresentata dal descrittore. Si osservi che ciò implica che tutti gli identificatori validi di PINO hanno una lunghezza inferiore a `MAX_ID_LEN`.

I prototipi delle funzioni da implementare sono riportati qui di seguito. Il comportamento di ciascuna funzione è descritto dal corrispondente commento.

```
struct descr_fun *ins_key_dfun(lista_dfun *pls, char *id);
/* aggiunge un nuovo descrittore per id alla lista *pls
 * ritorna il puntatore al nuovo descrittore inserito
 * o NULL in caso di errore o nel caso in cui id e'
 * gia' presente in *pls
 */

struct descr_fun *search_key_dfun(lista_dfun ls, char *id);
/* cerca il descrittore di funzione di id in ls
 * ritorna il puntatore al descrittore trovato
 * o NULL se non presente
 */

struct descr_var *ins_key_dvar(lista_dvar *pls, char *id);
/* aggiunge un nuovo descrittore per id alla lista *pls
 * ritorna il puntatore al nuovo descrittore inserito
 * o NULL in caso di errore o nel caso in cui id e'
 * gia' presente in *pls
 */

struct descr_var *search_key_dvar(lista_dvar ls, char *id);
/* cerca il descrittore di variabile di id in ls
 * ritorna il puntatore al descrittore trovato
 * o NULL se non presente
 */
```

Le funzioni della libreria `string.h`

Per realizzare la ricerca nelle tavole si consiglia di studiare le funzioni di manipolazione di stringhe contenute nella libreria `string.h` (si veda ad esempio il D&D). Alcune funzioni di tale libreria sono sicuramente utili per scrivere le funzioni che confrontano stringhe. Alcune funzioni della `string.h` potranno risultare utili anche per altri moduli del progetto.

1.3 Verifica

Per verificare le tavole dei simboli si utilizzerà il main file `pino-I.c` riportato in Appendice A.

Se il nome del file con il codice del modulo è `tavole.c`, su di un sistema linux con compilatore gcc il comando per la compilazione del programma di test è:

```
gcc -g pino-I.c pinodefs.c tavole.c -o pino-I
```

Come risultato della compilazione si ottiene l'eseguibile `pino-I` che, dopo aver chiesto di selezionare mediante una maschera la tavola da verificare, legge una sequenza di stringhe alfanumeriche da inserire e cercare all'interno della tavola selezionata.

A Modulo I: Verifica delle tavole dei simboli (TS)

```
/* -*- Mode: C -*- */
/* Time-stamp:  <pino-I.c 03/06/19 22:45:50 guerrini@zambujero.lan.home> */

/*********************************************************
* Progetto di Laboratorio di Programmazione
 * Stefano Guerrini - AA 2002-03
 *
 * Main per la verifica del modulo I: le tavole dei simboli (TS)
```

```
 * Per la compilazione servono i seguenti file:
 *   pinodefs.h        header principali defs
 *   pinodefs.c        alcune var globali e funzioni di utilita
 *   pino-I.c          questo file
 *   tavole.c          tavole dei simboli - modulo I (TS)
 * Per la compilazione su linux con gcc
 *   gcc -g pino-I.c pinodefs.c tavole.c -o pino-I
 * crea l'eseguibile pino-I con le info necessarie per il debugging
 ************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "pinodefs.h"

void get_word(char *str, unsigned lim) {
    /* legge una sequenza di caratteri alfanumerici
    * di lunghezza inferiore a lim e la memorizza in str */
    int c;
    while (--lim > 0 && (c = getchar()) != EOF && isalnum(c))
        *(str++) = c;
    *str = '\0';
    // elimina gli spazi bianchi dopo la parola fino all'eventuale newline
    while (c != '\n' && isblank(c))
        c = getchar();
}

void test_key_tavola(struct tavola *ptv) {
    /* verifica tavola implementata con vettore
     * - legge una stringa e prova ad inserirla nella tavola,
     *   in caso di inserimento verifica che indice di ritorno
     *   sia quello della stringa inserita stampandola
     * - prova nuovamante l'inserimento
     *   ci si aspetta che la risposta sia elemento gia' inserito
     *   o che la tavola e' piena
     * - verifica la ricerca
     * termina quando si immette la stringa vuota
     */
    int ind;
    char str[MAX_ID_LEN];
    printf("** Verifica funzioni tavola implementata con vettore **\n");
    init_tavola(ptv);
    printf("Dopo inizializzazione: disponibili %02d posti nella tavola.\n",
           avail_tavola(ptv));
    printf("Inserisci delle stringhe di lungh. max %02d.\n", MAX_ID_LEN);
    printf("[Stringa vuota per finire]\n");
    do {
        printf("tv> ");
        get_word(str, MAX_ID_LEN);
        if (*str == '\0')
            break; // Stringa vuota. Esci dal ciclo
        /* Prova ad inserire la stringa */
        printf("(ins  1) ");
        if ((ind = ins_key_tavola(ptv, str)) < 0)
            printf(avail_tavola(ptv) > 0 ?
                   "Gia' inserito!\n" : "Tavola piena!\n");
        else
            printf("'%s'\n", ptv->keys[ind]);
```

```
        /* Prova ad inserire la stringa una seconda volta */
        printf("(ins  2) ");
        if ((ind = ins_key_tavola(ptv, str)) < 0)
            printf(avail_tavola(ptv) > 0 ?
                   "Gia' inserito!\n" : "Tavola piena!\n");
        else
            printf("Errore: ci si aspetta -1 da ins_key_tavola!\n");
        /* Cerca la stringa */
        printf("(search) ");
        if ((ind = search_key_tavola(ptv, str)) < 0)
            printf(avail_tavola(ptv) > 0 ?
                   "Gia' inserito!\n" : "Tavola piena!\n");
        else
            printf("'%s'\n", ptv->keys[ind]);
        /* Stampa info sullo stato della tavola */
        printf("Inseriti %02d elementi\nDisponibili %02d posti\n",
               ptv->n, avail_tavola(ptv));
    } while (-1);
}

void test_tav_dfun(lista_dfun ldf) {
    /* verifica tavola descrittori funzione
     * opera sulla lista di descrittori di funzione ldf
     * che si suppone gia' inizializzata
     * - legge una stringa e prova ad inserirla nella lista
     *   in caso di inserimento verifica che il descrittore ottenuto
     *   come valore di ritorno sia quello giusto stampandone l'id
     * - prova nuovamante l'inserimento
     *   ci si aspetta che la risposta sia elemento gia' inserito
     * - verifica la ricerca dell'elemento appena inserito
     * termina quando si immette la stringa vuota
     */
    char str[MAX_ID_LEN];
    struct descr_fun *pdf;
    printf("** Verifica funzioni tavola descrittori funzione **\n");
    printf("Inserisci delle stringhe di lungh. max %02d.\n", MAX_ID_LEN);
    printf("[Stringa vuota per finire]\n");
    do {
        printf("df> ");
        get_word(str, MAX_ID_LEN);
        if (*str == '\0')
            break; // Stringa vuota. Esci dal ciclo
        /* Prova ad inserire la stringa */
        printf("(ins  1) ");
        if ((pdf = ins_key_dfun(&ldf, str)) == NULL)
            printf("Gia' inserito!\n");
        else
            printf("'%s'\n", pdf->id);
        /* Prova ad inserire la stringa una seconda volta */
        printf("(ins  2) ");
        if ((pdf = ins_key_dfun(&ldf, str)) == NULL)
            printf("Gia' inserito!\n");
        else
            printf("Errore: la stringa sarebbe dovuta essere nella lista!\n");
        /* Prova la ricerca della stringa, avendola appena inserita
         * mi aspetto sia presente nella tavola */
        printf("(search) ");
```

```
        pdf = search_key_dfun(ldf, str);
        printf("'%s'\n", pdf == NULL ?
               "Errore: la stringa sarebbe dovuta essere nella lista!\n" :
               pdf->id);
    } while (-1);
}

void test_tav_dvar(lista_dvar ldv) {
    /* verifica tavola descrittori variabile
     * opera sulla lista di descrittori di varaibile ldv
     * che si suppone gia' inizializzata
     * - legge una stringa e prova ad inserirla nella lista
     *   in caso di inserimento verifica che il descrittore ottenuto
     *   come valore di ritorno sia quello giusto stampandone l'id
     * - prova nuovamante l'inserimento
     *   ci si aspetta che la risposta sia elemento gia' inserito
     * - verifica la ricerca dell'elemento appena inserito
     * termina quando si immette la stringa vuota
     */
    char str[MAX_ID_LEN];
    struct descr_var *pdv;
    printf("** Verifica funzioni tavola descrittori variabili **\n");
    printf("Inserisci delle stringhe di lungh. max %02d.\n", MAX_ID_LEN);
    printf("[Stringa vuota per finire]\n");
    do {
        printf("dv> ");
        get_word(str, MAX_ID_LEN);
        if (*str == '\0')
            break; // Stringa vuota. Esci dal ciclo
        /* Prova ad inserire la stringa */
        printf("(ins  1) ");
        if ((pdv = ins_key_dvar(&ldv, str)) == NULL)
            printf("Gia' inserito!\n");
        else
            printf("'%s'\n", pdv->id);
        /* Prova ad inserire la stringa una seconda volta */
        printf("(ins  2) ");
        if ((pdv = ins_key_dvar(&ldv, str)) == NULL)
            printf("Gia' inserito!\n");
        else
            printf("Errore: la stringa sarebbe dovuta essere nella lista!\n");
        /* Prova la ricerca della stringa, avendola appena inserita
         * mi aspetto sia presente nella tavola */
        printf("(search) ");
        pdv = search_key_dvar(ldv, str);
        printf("'%s'\n", pdv == NULL ?
               "Errore: la stringa sarebbe dovuta essere nella lista!\n" :
               pdv->id);
    } while (-1);
}

int main(void) {
    /* Chiede, con una maschera, di selezionarte la tavola da verificare
     * Quindi, legge una sequenza di stringhe alfanumeriche di lunghezza
     * inferiore a MAX_ID_LEN e prova inserimento e ricerca nelle tavola
     * selezionata.
     */
```

```
    int c;
    struct tavola tv; // per il test della tavola implem con vettore
    lista_dfun ldf = NULL; // per il test della lista descr funzione
    lista_dvar ldv = NULL; // per il test della lista descr variabile
    do {
        printf("Digitare\n");
        printf(" 1 per il test delle tavole mediante vettori (tv)\n");
        printf(" 2 per il test delle liste descr. funzioni (df)\n");
        printf(" 3 per il test delle liste descr. variabili (tv)\n");
        printf(" 0 per terminare: ");
        /* leggi il primo carattere della riga ed ignora il resto */
        while ((c = getchar()) < '0' || c > '3');
        while (getchar() != '\n'); // elimina il resto della riga
        putchar('\n');
        switch (c) {
        case '0':
            exit(0); // termina
            break;
        case '1': /* Prova tavola mediante vettore */
            test_key_tavola(&tv);
            break;
        case '2': /* Prova tavola/lista descrittori di funzione */
            test_tav_dfun(ldf);
            break;
        case '3': /* Prova tavola/lista descrittori di funzione */
            test_tav_dvar(ldv);
            break;
        }
        putchar('\n');
    } while (-1);
}
```