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

Modulo V: Lo stack (ST)

Stefano Guerrini

A.A. 2002/03 – Canale P-Z
Versione del 24 giugno 2003

1 Modulo V: Lo stack (ST)

Gli elementi nello stack o pila che si deve implementare sono puntatori generici, quindi di tipo `void *`. Lo stack deve essere implementato mediante un vettore di memoria pari al numero massimo di elementi memorizzabili nello stack. Per questo motivo in `pinodefs.h` è definita la seguente struttura

```
/* Struttura per l'implementazione di uno stack di puntatori
 * mediante un vettore di memoria
 */
struct stack {
    size_t size; // dimensione dello stack (num max di puntatori nello stack)
    size_t nump; // numero di puntatori memorizzati nello stack
    void **vect; // vettore di memoria per memorizzare i puntatori,
                 // viene allocato dalla init_stack
};
```

Prima di poter essere utilizzato, uno stack va inizializzato, assegnandogli una dimensione ed allocando la memoria necessaria a contenere gli elementi (il campo `vect` della struttura). L'inizializzazione dello stack è eseguita dalla funzione

```
void *init_stack(struct stack *pst, size_t n);
/* Inizializza lo stack di puntatori *pst
 * allocando il vettore di memoria dello stack per
 * contenere sino a n puntatori
 * Ritorna un puntatore alla base del vettore di memoria
 * che implementa lo stack, o NULL in caso di errore
 */
```

Oltre ai campi `vect` e `size`, la struttura che implementa lo stack contiene il campo `nump` che indica il numero di elementi nello stack.

Per inserire ed eliminare elementi dallo stack si devono implementare le funzioni

```
void *push(struct stack *pst, unsigned n);
/* Inserisce n nuovi puntatori nelo stack *pst
 * Il valore dei puntatori inseriti nello stack e' indefinito,
 * in pratica, crea solo lo spazio per scrivere n nuovi puntatori
```

```
 * in testa allo stack
 * Ritorna il puntatore all'elemento del vettore in testa allo stack
 * dopo l'inserimento, o NULL in caso di errore (stack overflow)
 */

void *pop(struct stack *pst, unsigned n);
/* Elimina n puntatori dallo stack *pst
 * Ritorna il puntatore all'elemento del vettore in testa allo stack
 * dopo la cancellazione, o NULL in caso di errore (se non ci sono n elementi
 * da cancellare)
 */
```

che, rispettivamente, creano lo spazio per n puntatori in cima alla pila e rimuovono n puntatori dalla pila. Si osservi che la push non inserisce nessun elemento in testa alla pila. Dopo l'esecuzione di una `push(pst, n)`, in testa alla pila ci sono n posti che contengono puntatori dal valore indefinito, il contenuto dovrà essere inserito dopo l'esecuzione della push per mezzo della funzione `top`.

L'ultima funzione da implementare è la

```
void *top(struct stack *pst, unsigned n);
/* Ritorna il puntatore all'elemento del vettore in posizione n-esima
 * rispetto alla testa dello stack, o NULL in caso di errore (se lo stack
 * contiene meno di n+1 elementi)
 */
```

che permette di prendere il puntatore all'elemento in posizione n rispetto alla testa della pila. Si osservi che la funzione non deve ritornare il contenuto dela posizione a distanza n dalla testa della pila (ad esempio, della testa della pila se $n = 0$), ma il puntatore alla posizione n della pila.

1.1 Verifica

Per verificare le funzioni che implementano lo stack 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 delle funzioni che implementano lo stack è `stack.c` il comando per la compilazione del programma di test è:

```
gcc -g pino-V.c pinodefs.c stack.c -o pino-V
```

Come risultato della compilazione si otterrà l'eseguibile `pino-V` che, attraverso una maschera, permette di inserire in una pila dei puntatori agli elementi di un vettore definito nel programma di test, di eliminare elementi dalla pila, di leggere il valore riferito in posizione n-esima rispetto alla testa della pila e di stampare il contenuto della pila (i valori puntati dagli elementi nella pila).

A Modulo V: Verifica dello stack (ST)

```
 /* -*- Mode: C -*- */
/* Time-stamp:  <pino-V.c 03/06/24 15:39:17 guerrini@zambujero.lan.home> */

/*************************************************************
* Progetto di Laboratorio di Programmazione
 * Stefano Guerrini - AA 2002-03
 *
 * Main per la verifica del modulo V: lo stack (ST)
 * Per la compilazione servono i seguenti file:
 *   pinodefs.h        header principali defs
 *   pinodefs.c        alcune var globali e funzioni di utilita
```

```
 *   pino-V.c          questo file
 *   stack.c           stack - modulo V (ST)
 * Per la compilazione su linux con gcc
 *   gcc -g pino-V.c pinodefs.c stack.c -o pino-V
 * crea l'eseguibile pino-V con le info necessarie per il debugging
 ************************************************************/

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

#define DIM 32 // dimensione stack

#define N 16 // per prendere dei numeri quasi casuali
int V[N];

char *keys = ".+-=*"; // comandi del programma di test

int main(void) {
    int c, n, **p, ok = -1;
    unsigned count = 0; // contatore modulo N dei numeri inseriti
    for (n = 0; n < DIM; n++) // inizializza vettore numeri
        V[n] = n;
    init_stack(&lvars_stack, DIM); // inizializza lo stack
    printf("Lo stack ha dimensione %d\n", DIM);
    printf("Digitare\n");
    printf(" + n per inserire i puntatori a n numeri nella pila\n");
    printf(" - n per eliminare i puntatori a n numeri dalla pila\n");
    printf(" = n per stampare il numero puntato dall'n-esimo elemento in testa alla pila\n");
    printf(" * per stampare i numeri contenuti nella pila\n");
    printf(" . per terminare: \n");
    do {
        printf("$ ");
        while (! index(keys, (c = getchar())));
        switch (c) {
        case '.':
            exit(0); // termina
            break;
        case '+':
            scanf("%d", &n); // numero elementi da inserire
            if (ok = (push(&lvars_stack, n) != NULL)) {
                while (n-- > 0) // memorizza n puntatori ad elementi di V
                    *((int **)top(&lvars_stack, n)) = V+(count++ % N);
                printf("OK! Lo stack contiene %d elementi\n",
                       lvars_stack.nump);
            } else
                printf("Stack overflow\n");
            break;
        case '-':
            scanf("%d", &n); // numero elementi da eliminare
            if (ok = (pop(&lvars_stack, n) != NULL))
                printf("OK! Lo stack contiene %d elementi\n",
                       lvars_stack.nump);
            else
                printf("Errore! Nello stack ci sono solo %d elementi\n",
                       lvars_stack.nump);
```

```
            break;
        case '=':
            scanf("%d", &n); // elemento de leggere
            if (ok = ((p = top(&lvars_stack , n)) != NULL)) {
                // stampa il valore puntato dal puntatore preso dallo stack
                printf("%d: %d\n", n, **p);
            }
            else
                printf("Errore! Nello stack ci sono solo %d elementi\n",
                       lvars_stack.nump);
            break;
        case '*':
            // stampa i valori puntati dagli elementi contenuti nello stack
            for (n = 0; n < lvars_stack.nump; n++) {
                printf("%d: %d\n", n, **(int **)top(&lvars_stack, n));
            }
            break;
        }
        while (getchar() != '\n'); // elimina il resto della riga
    } while (-1);
}
```