Liste, Stack e Code

Andrea Marin

Università Ca' Foscari Venezia
Laurea in Informatica
Corso di Programmazione

a.a. 2012/2013

Section 1

Stack e liste

Funzionamento basilare

- Lo stack adotta una politica LIFO per la gestione dei dati
- Possiamo implementare uno stack usando le liste dove:
  - L'operazione di `push` corrisponde ad un inserimento in testa `prepend`
  - L'operazione di `pop` corrisponde ad un prelievo dalla testa
  - L'operazione `is_empty` testa il caso di lista vuota

Tipi e creazione di uno stack vuoto

```
struct cellint{
    int item;
    struct cellint *next;
};

typedef struct cellint *stack_int;

stack_int empty_stack() {
    return NULL;
}
```

Operazione di push

```
int push(stack_int *pstack, int elem) {
    stack_int newcell;
    newcell = (stack_int) malloc(sizeof(struct cellint));
    if (newcell) {
        newcell->item = elem;
        newcell->next = *pstack;
        *pstack = newcell;
        return 1;
    } else
        return 0;
}
```

Operazione di pop

```
int pop(stack_int *pstack, int *pelem) {
    if (*pstack) { /*stack non vuoto*/
        stack_int pc = *pstack;
        *pstack = pc->next;
        *pelem = pc->item;
        free(pc);
        return 1;
    } else
        return 0;
}
```

Test di stack vuoto

```
int is_empty(stack_int stack) {
    return (stack==NULL);
}
```

Section 2

Uno stack di stringhe

Memorizzazione delle stringhe nello stack

Tre possibili soluzioni con effetti diversi:

- Soluzione 1: si definisce una stringa con una lunghezza massima istanziata nella cella
- Soluzione 2: si definisce un puntatore ad una stringa non *controllata* dallo stack
  - Attenzione, soluzione rischiosa!
- Soluzione 3: si definisce un puntatore ad una stringa che sarà controllata dallo stack
  - Soluzione sicura

Soluzione 1 - tipi

```
#define MAXDIM 100

struct cellstr{
    char str[MAXDIM];
    struct cellstr *next;
};

typedef struct cellstr *stack_str;
```

Soluzione 1 - push

```
int push_str(stack_str *pstack, char *str){
    if (strlen(str)<MAXDIM) {
        stack_str pc=(stack_str) malloc(sizeof (struct cellstr));
        if (pc) {
            pc->next = *pstack;
            strcpy(pc->str, str);
            *pstack = pc;
            return 1;
        } else
            return 0;
    } else
        return 0;
}
```

Soluzione 1 - pop

```
int pop_str(stack_str *pstack, char *str) {
    if (*pstack != NULL) {
        stack_str pc = *pstack;
        strcpy(str, (*pstack)->str);
        *pstack = (*pstack)->next;
        free(pc);
        return 1;
    } else
        return 0;
}
```

Vantaggi/svantaggi della soluzione 1

Vantaggi

- Semplicità nella codifica della soluzione

Svantaggi

- Si deve prefissare una dimensione massima per le stringhe gestibili
- Spreco dello spazio in memoria per liste di stringhe brevi

Soluzione 2 - tipi

```
struct cellstr{
    char *str;
    struct cellstr *next;
};

typedef struct cellstr *stack_str;
```

Soluzione 2 - push

```
int push_str(stack_str *pstack, char *string){
    stack_str pc=(stack_str) malloc(sizeof (struct cellstr));
    if (pc) {
        pc->next = *pstack;
        pc->str = string;
        *pstack = pc;
        return 1;
    } else
        return 0;
}
```

Soluzione 2 - pop

```
int pop_str(stack_str *pstack, char **string) {
    if (*pstack != NULL) {
        stack_str pc = *pstack;
        *string = (*pstack)->str;
        *pstack = (*pstack)->next;
        free(pc);
        return 1;
    } else
        return 0;
}
```

Uso sbagliato della soluzione 2 - cosa stampa?

```
in main() {
    char buffer[100];
    stack_str pstack = NULL;
    int i;
    for (i=0; i<10; i++) {
        scanf(``%s'', buffer);
        push(&pstack, buffer);
    }
    char *restr;
    for (i=0; i<10; i++) {
        pop(&pstack, &restr);
        printf(``%s \n'', restr);
    }
    return 0;
}
```

Uso sbagliato della soluzione 2 - caso ancora peggiore

```
in main() {
    stack_str pstack = NULL;
    int i;
    for (i=0; i<10; i++) {
        char buffer[100];
        scanf(``%s'', buffer);
        push(&pstack, buffer);
    }
    char *restr;
    for (i=0; i<10; i++) {
        pop(&pstack, &restr);
        printf(``%s \n'', restr);
    }
    return 0;
}
```

Soluzione 2: vantaggi/svantaggi

Vantaggi:

- Semplicità nella codifica
- Minimizzazione dello spazio necessario

Svantaggi

- Bisogna tener presente che stiamo definendo una lista di puntatori a carattere e non di stringhe.
- La gestione della memorizzazione delle stringhe è delegata all'utente
- Possibilità di puntatori orfani

Soluzione 3 - definizione dei tipi

- Identici alla soluzione 2

```
struct cellstr{
    char *str;
    struct cellstr *next;
};

typedef struct cellstr *stack_str;
```

Soluzione 3 - push

```
int push_str(stack_str *pstack, char *string){
    stack_str pc=(stack_str) malloc(sizeof (struct cellstr));
    if (pc) {
        pc->next = *pstack;
        pc->str = (char*)malloc(strlen(string)+1);
        if (!pc->str) { /*Allocazione stringa fallita*/
            free(pc); /*libera lo spazio allocato per la cella*/
            return 0;
        } else {
            strcpy(pc->str, string);
            *pstack = pc;
            return 1;
        }
    } else
        return 0;
}
```

Soluzione 3 - pop

```
int pop_str(stack_str *pstack, char **string) {
    if (*pstack != NULL) {
        stack_str pc = *pstack;
        *string = (*pstack)->str;
        *pstack = (*pstack)->next;
        /*We do not deallocate the string!*/
        free(pc);
        return 1;
    } else
        return 0;
}
```

Soluzione 3 - flush (svuota) dello stack

```
void flush(stack_str *pstack) {
    strack_str pc;
    while (*pstack) {
        pc = *pstack;
        *pstack = pc->next; /*scorre la lista*/
        free(pc->str); /*libera lo spazio della stringa*/
        free(pc); /*libera lo spazio della cella*/
    }
}
```

Soluzione 3: vantaggi/svantaggi

Vantaggi:

- Uso sicuro delle strutture dati
- Semplicità nell'uso
- Gestione di stringhe di lunghezza arbitraria

Svantaggi:

- Maggiore richiesta di memoria rispetto alla soluzione 2
- Salvo casi particolari, minor uso della memoria rispetto alla soluzione 1