Laboratorio di Elementi di Architetture e Sistemi Operativi

Soluzioni degli Esercizi del 9 Maggio 2012

Esercizio 1. *Scrivere un programma che:*

1. *legga a da tastiera un vettore di float di dimensione non nota e introdotta da tastiera;*
2. *calcoli la somma degli elementi del vettore;*
3. *stampi a schermo la somma.*

Utilizzare i puntatori e l'allocazione dinamica della memoria per fare in modo che il programma non debba assumere nessuna limitazione sulla lunghezza del vettore.

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

int main()
{
    float *v, somma = 0;
    int i, n;

    printf("Inserire il numero di elementi del vettore: ");
    scanf("%d", &n);

    v = (float *)malloc(n*sizeof(float));
    if(v == NULL) {
        fprintf(stderr, "Errore nell'allocazione della memoria!\n");
        return 1;
    }
    for(i = 0; i < n; i++)
    {
        printf("Elemento %d: ",i);
        scanf("%f", &v[i]);
    }

    for(i = 0; i < n; i++)
    {
        somma += v[i];
    }

    printf("La somma degli elementi è: %lf\n",somma);

    return 0;
}
```

Esercizio 2. *Scrivere un programma che:*

1. *legga a da tastiera una matrice $n \times m$ di float. n ed m non sono noti a priori e sono introdotti da tastiera;*
2. *genera la matrice trasposta e la stampi a schermo.*

Utilizzare i puntatori e l'allocazione dinamica della memoria per fare in modo che il programma non debba assumere nessuna limitazione sulla dimensione della matrice.

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

```
float *trasposta(float *a, int n, int m) {
    int i, j;
    float *b;

    b = (float *)malloc(m * n * sizeof(float));
    if(b != NULL) {
        for(i = 0; i < n; i++) {
            for(j = 0; j < m; j++) {
                b[j*n+i] = a[i*m+j];
            }
        }
    }
    return b;
}

int main()
{
    float *a, *b;
    int i, j, n, m;

    printf("Inserire il numero di righe della matrice: ");
    scanf("%d", &n);
    printf("Inserire il numero di colonne della matrice: ");
    scanf("%d", &m);

    a = (float *)malloc(n*m*sizeof(float));
    if(a == NULL) {
        fprintf(stderr, "Errore nell'allocazione della memoria!\n");
        return 1;
    }

    for(i = 0; i < n; i++)
    {
        for(j = 0; j < m; j++) {
            printf("Elemento (%d,%d): ", i, j);
            scanf("%f", &a[i*m+j]);
        }
    }

    b = trasposta(a, n, m);
    if(b == NULL) {
        fprintf(stderr, "Errore nell'allocazione della memoria!\n");
        return 1;
    }

    printf("Matrice trasposta:\n");
    for(i = 0; i < m; i++)
    {
        for(j = 0; j < n; j++) {
            printf("%lf\t", b[i*n+j]);
        }
        putchar('\n');
    }

    return 0;
```

```
}
```

Esercizio 3. *Si scriva un programma in linguaggio C che effettui le seguenti operazioni:*

- *tramite la funzione* `leggi()`*, legga da terminale i dati di* `n` *studenti (con* `n` *inserito da tastiera) costituiti da Nome, Cognome, Voto, e li inserisca in un vettore, rappresentando i dati di ogni studente con una struttura;*
- *calcoli con la funzione* `media()` *la media dei voti;*
- *stampi a terminale i nominativi di ciascuno studente;*
- *stampi poi a terminale il voto medio.*

Per semplificare lo svolgimento, si assuma che il numero di studenti non sia superiore a 20*, e che la lunghezza di nome e cognome non superi i* 20 *caratteri ciascuno.*

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

#define MAXCHAR 20

typedef struct stud {
    char nome[MAXCHAR+1];
    char cognome[MAXCHAR+1];
    int voto;
} studente;

studente *leggi(int *pn) {
    studente *v;
    int i;
    printf("Inserire il numero di studenti: ");
    scanf("%d", pn);
    v = (studente *)malloc((*pn)*sizeof(studente));
    if(v != NULL) {
        for(i = 0; i < *pn; i++) {
            printf("Studente %d\n",i+1);
            printf("Cognome: ");
            scanf("%s", v[i].cognome);
            printf("Nome: ");
            scanf("%s", v[i].nome);
            printf("Voto: ");
            scanf("%d", &v[i].voto);
            printf("\n");
        }
    }
    return v;
}

float media(studente *v, int n) {
    float somma = 0;
    int i;
    for(i = 0; i < n; i++) {
        somma += v[i].voto;
    }
    return somma/n;
}

void stampanomi(studente *v, int n) {
    int i;
```

```
    for(i = 0; i < n; i++) {
        printf("%s %s\n",v[i].cognome,v[i].nome);
    }
}

int main()
{
    studente *v;
    int n;

    v = leggi(&n);
    if(v == NULL) {
        fprintf(stderr, "errore di allocazione della memoria!\n");
        return 1;
    }
    stampanomi(v, n);
    printf("Voto medio: %lf\n", media(v,n));

    return 0;
}
```

Esercizio 4.

1. *Modificare il codice per la gestione delle liste visto a lezione per gestire una pila (stack) di* stringhe. *In particolare, si implementino le operazioni di inserimento (`pop`) ed estrazione (`push`) di una stringa dalla pila.*
2. *Usare le funzioni `push` e `pop` per implementare un programma `ribalta` che scriva le righe di un testo dall'ultima alla prima. Il programma legge l'input da un file passato come* primo argomento *del `main`, e stampa il risultato sullo schermo. Si supponga che le linee del testo di input non siano mai più lunghe di 80 caratteri.*

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

#define MAXCHAR 80

typedef struct elem {
  char *key;
  struct elem *next;
} elemen_t;

elemen_t* head = NULL;

int is_empty(void) {
  return(head == NULL);
}

void push(char *key) {
  elemen_t *paux;
  paux = (elemen_t *)malloc(sizeof(elemen_t));
  if(paux != NULL) {
    paux->key = key;
    paux->next = head;
    head = paux;
  }
}
```

```
char *pop(void) {
  elemen_t *paux;
  char *key = NULL;
  if(head != NULL) {
    paux = head;
    head = head->next;
    key = paux->key;
    free(paux);
  }
  return key;
}

void togliacapo(char *s) {
  for( ; *s != '\0' && *s != '\n'; s++) ;
  *s = '\0';
}

int main(int argc, char *argv[])
{
  char *res, *riga;
  FILE *input;

  if(argc != 2) {
    fprintf(stderr, "Uso: ribalta nomefile\n");
    return 1;
  }
  input = fopen(argv[1], "r");
  if(input == NULL) {
    fprintf(stderr, "errore nell'apertura del file %s\n", argv[1]);
    return 1;
  }
  riga = (char *) malloc(MAXCHAR+1);
  if(riga == NULL) {
    fprintf(stderr, "errore di allocazione della memoria!\n");
    return 1;
  }
  res = fgets(riga,MAXCHAR,input);
  while(res != NULL) {
    togliacapo(riga);
    push(riga);
    riga = (char *) malloc(MAXCHAR+1);
    if(riga == NULL) {
      fprintf(stderr, "errore di allocazione della memoria!\n");
      return 1;
    }
    res = fgets(riga,MAXCHAR,input);
  }
  if(fclose(input) != 0) {
    fprintf(stderr, "Errore nella chiusura del file %s\n", argv[1]);
    return 1;
  }

  while(!is_empty()) {
    riga = pop();
```

```
    puts(riga);
    free(riga);
  }

  return 0;
}
```