Liste

Esercizio: gestione di un testo

Dato un file di testo, trasformarlo spezzando tutte le righe di lunghezza superiore a dim in sequenze di righe di lunghezza dim

Trasformare poi il testo in una sequenza di righe tutte di dimensione dim, tranne, eventualmente, l'ultima

Gestione testo tramite liste

```
#include <stdio.h>
#include <stdlib.h>
typedef int bool;
#define TRUE  1
#define FALSE 0

struct StructCar {                          /* rappresentazione della lista di caratteri */
  char car;
  struct StructCar *next_car;          /* puntatore al prossimo carattere */
};

typedef  struct StructCar TipoCar;
typedef  TipoCar *TipoPuntCar;

struct StructRiga {                         /* rappresentazione della lista di righe */
  TipoPuntCar inizio;                       /* puntatore iniziale alla lista di caratteri */
  struct StructRiga *next_riga;       /* puntatore alla riga successive */
};

typedef  struct StructRiga TipoRiga;
typedef  TipoRiga *TipoPuntRiga;

typedef  TipoPuntRiga TipoTesto;
```

Gestione testo tramite liste

```
void SpezzaRiga(TipoPuntRiga riga, int dim)
  /* Spezza una riga in una parte di lunghezza dim ed in un resto. */
{
  TipoPuntCar car_corr;              /* puntatore al car. corrente nella riga corrente */
  TipoPuntRiga resto;                /* puntatore usato per creare la riga con il resto della riga spezzata */
  int num_car;                       /* contatore del numero di caratteri nella riga corrente */

  car_corr = riga->inizio;
  if (car_corr != NULL) {                    /* la riga non è vuota */
    for (num_car = 1;
        car_corr->next_car != NULL && num_car < dim;
                    /* la riga corrente non è terminata e non si è ancora raggiunto il carattere in posizione dim */
        num_car++)
              car_corr = car_corr->next_car;

    if (car_corr->next_car != NULL) {          /* la riga corrente viene spezzata */
      resto = malloc(sizeof(TipoRiga));        /* alloca il nodo iniziale per una nuova riga */
      resto->inizio = car_corr->next_car;      /* la nuova riga contiene il resto della riga spezzata */
      car_corr->next_car = NULL;               /* chiude la riga spezzata */

        /* inserisce la riga con il resto prima della prossima riga di testo */
      resto->next_riga = riga->next_riga;
      riga->next_riga = resto;
    }
  }
}  /* SpezzaRiga */
```

Gestione testo tramite liste

```
void RidimensionaTesto(TipoTesto *testo, int dim)
  /* Trasforma testo spezzando tutte le righe di lunghezza superiore a dim in
     sequenze di righe di lunghezza dim. */
{
  TipoPuntRiga riga_corr;

  riga_corr = *testo;
  while (riga_corr != NULL) {
    SpezzaRiga(riga_corr, dim);                  /* spezza la riga corrente */
    riga_corr = riga_corr->next_riga;  /* si posiziona sulla riga successiva */
  }
}  /* RidimensionaTesto */
```

Gestione testo tramite liste

```
void CancellaRigheIniziali Vuote(TipoTesto *testo)
  /* Cancella da testo tutte le righe iniziali vuote. */
{
  TipoPuntRiga raux;
  bool continua = TRUE;        /* indica se si deve continuare a cancellare */

  while (*testo != NULL && continua) {
    if ((*testo)->inizio == NULL) {                    /* la riga è vuota */
      raux = *testo;
      *testo = (*testo)->next_riga;
      free(raux);
    }
    else   /* si è arrivati ad una riga non vuota e si smette di cancellare */
      continua = FALSE;
  }
}  /* CancellaRigheIniziali Vuote */
```

Gestione testo tramite liste

```
void UnicaRiga(TipoTesto *testo)
  /* Trasforma testo collegando tutte le righe in un'unica riga. */
{
  TipoPuntCar car_corr;                 /* puntatore al carattere corrente  */
  TipoPuntRiga riga_succ;               /* ptr alla riga che segue quella con il carattere corrente */
  TipoPuntRiga raux;

  CancellaRigheInizialiVuote(testo);
  if (*testo != NULL) {                                         /* testo punta alla prima riga non vuota */
          car_corr = (*testo)->inizio;
          riga_succ = (*testo)->next_riga;
          while (riga_succ != NULL) {                           /* esistono ancora righe successive */
                  while (car_corr->next_car != NULL) {
                                  /* non si e` ancora alla fine della riga */
                          car_corr = car_corr->next_car;
                  }
                  car_corr->next_car = riga_succ->inizio;  /* collega la riga successive in coda a quella corrente  */

                /* cancella il nodo di riga relativo alla riga successiva */
                  raux = riga_succ;
                  riga_succ = riga_succ->next_riga;
                  free(raux);
          }
          (*testo)->next_riga = NULL;          /* testo termina dopo la prima riga */
}
}  /* UnicaRiga */
```

Gestione testo tramite liste

```
void CompattaTesto(TipoTesto *testo, int dim)
  /* Trasforma testo in una sequenza di righe tutte di dimensione dim, tranne
     eventualmente l'ultima. */
{
  UnicaRiga(testo);
  RidimensionaTesto(testo, dim);
}  /* CompattaTesto */
```

Gestione testo tramite liste

```
bool LeggiRiga(FILE *fi, TipoPuntCar *inizio_riga)
  /* Legge una riga di testo dal file fi e restituisce in *inizio_riga il puntatore iniziale alla lista di caratteri della
riga letta. Restituisce FALSE se si è giunti alla fine del file e la riga letta è vuota.  */
{
  TipoPuntCar caux;
  char ch;

  *inizio_riga = malloc(sizeof(TipoCar));
  caux = *inizio_riga;
  ch = getc(fi);
  while (ch != '\n' && ch != EOF) {
    caux->next_car = malloc(sizeof(TipoCar));
    caux = caux->next_car;
    caux->car = ch;
    ch = getc(fi);
  }
  caux->next_car = NULL;
  caux = *inizio_riga;
  *inizio_riga = (*inizio_riga)->next_car;
  free(caux);
  return (ch != EOF || *inizio_riga != NULL);
}  /* LeggiRiga */
```

Gestione testo tramite liste

```
void LeggiTesto(char *nomefile, TipoTesto *testo)
  /* Restituisce un testo le cui righe vengono lette dal file di nome nomefile. Ogni riga è costituita dalla lista dei
caratteri della corrispondente riga nel file. */
{
  TipoPuntRiga raux;
  TipoPuntCar caux;
  FILE *fi;

  fi = fopen(nomefile, "r");
  if (fi == NULL) {
    fprintf(stderr, "Errore nell'apertura in lettura di %s", nomefile);
    exit (1);
  }

  *testo = malloc(sizeof(TipoRiga));
  raux = *testo;
  while (LeggiRiga(fi, &caux)) {
    raux->next_riga = malloc(sizeof(TipoRiga));
    raux = raux->next_riga;
    raux->inizio = caux;
  }
  raux->next_riga = NULL;
  raux = *testo;
  *testo = (*testo)->next_riga;
  free(raux);
  fclose(fi);
}  /* LeggiTesto */
```

Gestione testo tramite liste

```
void ScriviRiga(FILE *fi, TipoPuntCar riga)
  /* Scrive riga sul file fi. */
{
  while (riga != NULL) {
    putc(riga->car, fi);
    riga = riga->next_car;
  }
  putc('\n', fi);
}  /* ScriviRiga */
```

Gestione testo tramite liste

```
void ScriviTesto(char *nomefile, TipoTesto testo)
  /* Scrive testo sul file di nome nomefile. */
{
  FILE *fi;

  fi = fopen(nomefile, "w");
  if (fi == NULL) {
    fprintf(stderr, "Errore nell'apertura in scrittura di %s", nomefile);
    exit (1);
  }

  while (testo != NULL) {
    ScriviRiga(fi, testo->inizio);
    testo = testo->next_riga;
  }
  fclose(fi);
}  /* ScriviTesto */
```

Gestione testo tramite liste

```
void CancellaRiga(TipoPuntCar *riga)
  /* Cancella riga liberando la memoria occupata. */
{
  TipoPuntCar caux;

  while (*riga != NULL) {
    caux = *riga;
    *riga = (*riga)->next_car;
    free(caux);
  }
}  /* CancellaRiga */


void CancellaTesto(TipoTesto *testo)
  /* Cancella testo liberando la memoria occupata. */
{
  TipoPuntRiga raux;

  while (*testo != NULL) {
    CancellaRiga(&(*testo)->inizio);
    raux = *testo;
    *testo = (*testo)->next_riga;
    free(raux);
  }
}  /* CancellaTesto */
```

Gestione testo tramite liste

```
int main(void)
{
  TipoTesto testo;
  int dim;

  do {
            printf("Immetti la lunghezza massima delle righe di testo: ");
            scanf("%d", &dim);
            if (dim <= 0)
              printf("La lunghezza massima deve essere un intero positivo.\n");
  } while (dim <= 0);
  LeggiTesto("dati.txt", &testo);
  RidimensionaTesto(&testo, dim);
  ScriviTesto("datiou1.txt", testo);
  CancellaTesto(&testo);
  LeggiTesto("dati.txt", &testo);
  CompattaTesto(&testo, dim);
  ScriviTesto("datiou2.txt", testo);
  CancellaTesto(&testo);
  return 0;
}  /* GestioneTesti */
```