Laurea In Ingegneria dell'Informazione

Esercitazioni Guidate di Tecniche della Programmazione

Note introduttive: Seguire le raccomandazioni date nelle precedenti EG …

Esercitazione Guidata 11, parte 1

liste concatenate *rappresentate mediante struct e puntatori*: ancora sull'inserimento in coda, eliminazione da lista, inserimento ordinato.

1.1. Inserimento in coda (`liste6.c`)

Si richiede di eseguire l'esercizio precedente (liste5 …) utilizzando la tecnica di inserimento in coda con l'uso del record generatore:

- `pgen` sia il puntatore al record generatore (una struttura fittizia che rimane vuota di informazione ma ha come nodo successore quello che alla fine sara' il primo nodeo della lista);
- inizialmente `ultimo=pgen`;
- poi si esegue un ciclo di inserimenti in coda, usando e aggiornando ultimo:
  - allocazione di un nodo puntato da `ultimo->next`
  - aggiornamento di ultimo in modo che punti al nuovo nodo allocato in coda
  - inserimento del dato in `ultimo->info`
  - lettura del prossimo dato (quello da inserire alla prossima iterazione, o quello che fara' terminare il ciclo di inserimenti).

1.2. "Inserimento in coda" 2 (`liste7.c`)

Eseguire l'esercizio precedente, ma definendo ed utilizzando una funzione apposita per eseguire gli inserimenti in coda, il cui prototipo sia il seguente:

```
int insCodaLista(TipoLista * plis, TipoElemLista elem);
```

questa funzione inserisce `elem` in coda alla lista; `plis` è l'indirizzo della variabile puntatore che punta all'inizio della lista. (Quindi si usa questa funzione per eseguire un inserimento in lista).

Ed ecco come viene usata:

```
  leggiElemLista(&el);        /* prima lettura */

  while (!ugualiElem(el, '\n')) {
       /* inserimento in coda del dato letto precedentemente */
    ins=insCodaLista(&list, el);

    if (ins==0) {  /* se c sono stati problemi usciamo dal ciclo */
      printf("\nproblemi in inserimento\n");
      break;
    }

    leggiElemLista(&el);       /* lettura elemento successivo */
  }
```

1.3. "Inserimento in coda" 3: i file (`liste8.c`)

Scrivere un programma che

- costruisca una lista con i dati (**interi**) letti da un file di testo
- stampi la lista

il nome del file di testo deve essere ricevuto da input

segue un suggerimento se serve …

Suggerimento 1 di 2:

ecco i prototipi delle funzioni usate nel programma del file `liste8.c`

```
void stampaLista(TipoLista lis);
void stampaElemLista(TipoElemLista v);
int insCodaLista(TipoLista * plis, TipoElemLista elem);

/* aggiunta per leggere da file gli elementi della lista */
void leggiElemListaDaFile(FILE * fin, TipoElemLista *pelem);

/* aggiunta per costruire la lista: la funzione riceve il nome di
un file di testo e produce una lista con i dati del file,
restituendo il puntatore all'inizio della lista */
TipoLista costruisciListaDaFile (char nmf[31]);
```

segue un suggerimento se serve …

Suggerimento 2 di 2:

ed ecco la funzione `main()` del programma nel file `liste8.c`

```
int main() {
  TipoLista list;
  TipoElemLista el;
  char nomeFile[31];

  printf(" - nome del file: ");
  scanf("%s", nomeFile);

  list = costruisciListaDaFile(nomeFile);

  printf(" - ecco la lista:\n");
  stampaLista(list);

printf("\nFINE\n");
return 0;
}
```

1.4. Eliminazione da lista (`liste9.c`)

Scrivere un programma che

- costruisca una lista con i dati (**interi**) letti da un file di testo
- stampi la lista
- per due volte
  chieda un intero da input e lo elimini dalla lista (se lo trova)
- stampi di nuovo la lista
- scarichi i dati in lista nello stesso file di testo

il nome del file di testo deve essere ricevuto da input.

Ovviamente si opera in modo da estendere il programma precedente, aggiungendo ed usando funzioni opportune.

Ecco le due funzioni principali aggiunte nel file `liste9.c`

```
   /* aggiunta per eliminare un elemento da una lista */
void eliminaDaLista(TipoLista * plis, TipoElemLista elem);

   /* aggiunta per scaricare i dati della lista lis nel file di nome nmf */
void listaInFile (TipoLista lis, char nmf[31]);
```

1.5. inserimento ordinato in lista (`liste10.c`)

Scrivere un programma in cui venga costruita e poi stampata una lista di interi, i cui elementi siano ordinati in senso crescente.

I dati da inserire in lista vengono forniti da input come una sequenza (disordinata) di interi, terminata da 0

La soluzione da adottare consiste nel costruire la lista mediante una serie di inserimenti ordinati. Ogni inserimento viene effettuata dia una chiamata all'opportuna funzione `insOrdLista()`. L'algoritmo di inserimento ordinato che si consiglia è quello che fa uso del record generatore e di due puntatori di appoggio.

Segue un suggerimento se serve

…

Suggerimento:

Ecco i prototipi delle funzioni necessarie (oltre a quelle immaginabili, gia' sviluppate in esercizi precedenti).

```
int precedeElem(TipoElemLista el1, TipoElemLista el2);
/* aggiunta per determinare se un certo elemento va inserito prima
o dopo di un altro: restituisce 1 se el1 precede el2 in ordinamento
crescente (0 altrimenti) */

int insOrdLista(TipoLista * plis, TipoElemLista elem);
/* aggiunta per eseguire l'inserimento: inserisece elem
ordinatamente in lista; plis e' l'indirizzo della variabile
puntatore che punta all'inizio della lista */
```