

Esercitazioni di Tecniche di Programmazione

due avvertenze:

- 1) Le soluzioni agli esercizi, le versioni di programmi dati nel testo delle esercitazioni e quant'altro sono raggiungibili tramite la pagina web del corso. Nel titolo di ogni sezione è specificato tra parentesi il nome del (o dei) file in cui è proposta una soluzione (se disponibile ...).
- 2) I programmi che scriveremo dovranno essere in accordo con la definizione standard **ANSI C** del linguaggio C; perciò, prima di cominciare vogliamo assicurarci che l'ambiente di programmazione che usiamo "usi" anche lui la medesima definizione.
 - a. SE si usa il Dev C++, nella versione 4.9.9.2 (lingua inglese) bisogna andare nel menu' "Tools", selezionare "Compiler Options", scegliere "Settings" e poi "C Compiler" (selezionare almeno "Support all ANSI Standard C Programs")
 - b. Se si usa il vecchio ed eroico Turbo C++, per essere sicuri di star usando ANSI C, bisogna assegnare opportunamente una certa opzione: aprire il menù *OPTIONS*, selezionare *Compiler* e poi *Source*. Nella finestra di scelta che appare, selezionate ANSI C.

5. Quinta esercitazione autoguidata: ancora sull'inserimento in coda in liste concatenate; eliminazione da lista; inserimento ordinato.

5.1. Inserimento in coda (LISTE6.C)

Si richiede di eseguire l'esercizio precedente 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 sarà il primo nodo della lista);
- inizialmente ultimo=pgen;
- poi si esegue un ciclo di inserimenti in coda, usando e aggiornando ultimo:
 - o allocazione di un nodo puntato da ultimo->next
 - o aggiornamento di ultimo in modo che punti al nuovo nodo allocato in coda
 - o inserimento del dato in ultimo->info
 - o lettura del prossimo dato (quello da inserire alla prossima iterazione, o quello che farà terminare il ciclo di inserimenti).

5.2. “Inserimento in coda” 2 (LISTE7.C)

Eeguire l’esercizio precedente, ma definendoo 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` e' 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 */
}
```

5.3. “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

Suggerimento:

ecco i prototipi delle funzioni usate nel programma 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]);
```

Suggerimento 2:

ed ecco la funzione main() del programma 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;
}
```

5.4. *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]);
```

5.5. *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 da una chiamata all'opportuna funzione `insOrdLista()`. L'algoritmo di inserimento ordinato che si consiglia e' quello che fa uso del record generatore e di due puntatori di appoggio.

Suggerimento:

Ecco i prototipi delle funzioni necessarie (oltre a quelle immaginabili, già 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: inserisce elem  
ordinatamente in lista; plis e' l'indirizzo della variabile  
puntatore che punta all'inizio della lista */
```