UNIVERSITÀ
degli STUDI
di CATANIA

Dipartimento
di Fisica
e Astronomia
"Ettore Majorana"

L-30 CLASSE DELLE LAUREE IN SCIENZE E TECNOLOGIE FISICHE
CORSO DI LAUREA IN FISICA
PROGRAMMAZIONE AD OGGETTI E BIG DATA

SANDRO FIORETTO

REALIZZAZIONE IN C DELL'OGGETTO LISTA

SOMMARIO

L'OBIETTIVO DELLA RELAZIONE È PRESENTARE L'OGGETTO "LISTA", REALIZZATO IN C IN DUE VERSIONI: UNA CLASSICA IN CUI VIENE CREATO UN NUOVO ELEMENTO AD OGNI VALORE INSERITO E UNA PIÙ VELOCE PERCHÉ UTILIZZA, ASSIEME A TUTTE LE ALTRE LISTE DELLO STESSO TIPO, UN UNICO SPAZIO IN MEMORIA PREALLOCATO.

ANNO ACCADEMICO 2022/2023 - 3° ANNO

1 Cos'è la lista in informatica

La lista è una struttura dati dinamica. A differenza dell'array può facilmente cambiare numero di elementi e i suoi elementi non sono necessariamente contigui in memoria.
La complessità delle seguenti funzioni della lista non dipende dalla sua dimensione:

- inserimento e rimozione in testa alla lista
- inserimento in coda alla lista

ma vi dipende nel caso di:

- rimozione in coda alla lista
- inserimento e rimozione all'interno della lista
- ricerca di elementi interni alla lista

È perciò particolarmente utile quando il numero di elementi non è noto a priori e non si devono effettuare molte operazioni con elementi interni alla lista.

2 Tipi di lista

Sono forniti due tipi di lista, chiamati di seguito list_dynamic e list_table.

La list_dynamic è la versione più classica della lista: la lista occupa tanta più memoria quanti sono i suoi elementi. Ha il vantaggio rispetto alla versione successiva di occupare solamente la memoria necessaria per contenere i suoi elementi, tuttavia l'inserimento di ciascun elemento richiede un'allocazione in memoria, il ché rende la list_dynamic più lenta degli array o della sua controparte list_table.

La list_table è una versione in cui viene preallocata una quantità di memoria arbitraria (d'ora in poi $table$) all'istanziamento della prima lista e nella quale vengono salvati gli elementi di tutte le liste della stessa natura. Ciascuna table infatti deve avere elementi della stessa dimensione quindi potrà essere condivisa solo da liste che che contengono lo stesso tipo di dato (sarà istanziata una table per le liste che contengono interi, una per i float etc.).
Il vantaggio di questa versione rispetto alla precedente è che lavorare con la table è più veloce che lavorare con puntatori; rispetto agli array è che le liste contenute nella table possono avere numero di elementi variabile.
Lo svantaggio è che quando la memoria preallocata si riempie la table deve essere ricopiata in una zona di memoria più grande, il ché rallenta temporaneamente il programma. Per decidere come ingrandire la table sono forniti due tipi di resize:

- resize_default: è il tipo di resize della table se non è stato specificato diversamente. La table si espande automaticamente quando si cerca di inserire elementi oltre la sua capienza.
- resize_manual: le funzioni di inserimento tornano errore se la tabella è già piena e questa va espansa manualmente attraverso la funzione $expand\_table$.

La dimensione della table in ogni caso può essere modificata manualmente attraverso le funzioni $expand\_table$ e $shrink\_table$, mentre il suo tipo di resize può essere scelto alla creazione della lista mediante $malloc\_list\_specify\_table$ o modificato successivamente con $change\_resize\_table$. AGGIUNGERE HREF ALLE RISPETTIVE SEZIONI DELLE FUNZIONI

2.1 Tipi di dati contenuti

I tipi che l'oggetto può contenere, stabiliti all'istanziamento di ciascuna lista, sono "CHAR", "INT", "LONG", "FLOAT", "DOUBLE", "GENERIC".
"GENERIC" è un tipo di dato con dimensione variabile individuato da un puntatore all'indirizzo di memoria in cui è contenuto e un intero unsigned in cui è salvata la sua dimensione, ottenibile ad esempio con la funzione standard di C $sizeof$.

È importante sottolineare che quando un elemento viene inserito all'interno della lista viene creata una sua copia locale. Ciò permette di mantenere invariato l'elemento salvato nella lista pur modificando la variabile attraverso cui è stato inserito (presa ad esempio in input da $insert\_first$).

Spesso è conveniente lavorare con array di dati in modo da non dover traversare la lista un elemento alla volta, ma N alla volta, con N numero di elementi di ciascun array. All'istanziamento della lista, è fornita la possibilità di scegliere il numero di valori contenuti in ciascuno dei suoi elemento. Si noti che questo numero è fissato all'istanziamento della lista e deve essere lo stesso per tutti gli array in essa contenuti.

Poiché C non permette l'overloading delle funzioni, sarebbero richieste tante funzioni di inserimento ed estrazione quanti sono i tipi contenuti nella lista. Piuttosto si è scelto di utilizzare il $type\ casting$ di C.
Le funzioni di inserimento e di estrazione prendono in input elementi di tipo ALL_TYPE, che è definito da un programma lanciato in fase di compilazione come il tipo più grande tra quelli contenibili dalla lista. Per evitare warning dovuti al casting delle variabili, è fornita una macro $TO\_ALLTYPE$, da utilizzare quando si passa una variabile qualsiasi a una funzione che ne aspetta una di tipo ALL_TYPE.
Ad esempio nel caso di inserimento ed estrazioni di valori di tipo float, si avranno le seguenti funzioni:

```
insert_first(plista, TO_ALLTYPE(f), ...)
extract_first(plista, TO_ALLTYPE(&f), ...)
```

Sono fornite due macro analoghe, TO_PVOID e $TO\_BASETYPE$, per passare da una variabile di tipo ALL_TYPE a una variabile rispettivamente di tipo puntatore oppure tipo base (int, float, etc.).
Queste sono particolarmente utili nel definire funzioni come $pcustom\_print$ e $pcustom\_compare$, che prendono in input variabili di tipo ALL_TYPE ma che possono essere riferite a elementi di tipo puntatore (ad esempio nel caso di generic) o di tipo base. AGGIUNGERE HREF ALLA SEZIONE SUCCESSIVA

3 Manuale d'uso della lista

Di seguito sono riportate le funzioni membro fornite e come si utilizzano.

3.1 Istanziamento della lista

La lista è istanziata mediante la funzione

```
pvoid malloc_list(type_list type_list,
                  pchar type_string,
                  unsi dim_array);
```

dove gli argomenti di input sono rispettivamente:

- type_list: tipo di lista da istanziare, da scegliere tra $type\_list\_dynamic$ e $type\_list\_table$, con le differenze riportate in Sezione 2
- type_string: è una stringa che corrisponde al tipo del dato da contenere. Deve essere una tra: "CHAR", "INT", "LONG", "FLOAT", "DOUBLE", "GENERIC". .....