Informatica Generale II

A.A. 2004/2005

Esame: 27 ottobre 2005

Compilatore C++2Java

Alla ditta InfGenStud è stato richiesto di realizzare un traduttore dal linguaggio C++ al linguaggio Java. Tale software riceverà in ingresso un programma S scritto in C++ e produrrà in uscita un corrispondente programma D in Java.

Nella realizzazione di tale software la ditta InfGenStud ha ritenuto necessario costruire un una "tabella dei simboli", ovvero il dizionario del programma di partenza: per ogni simbolo del programma di partenza (ad esempio il nome di una variabile) la tabella dei simboli associa informazioni dell'oggetto avente quel nome (ad esempio il suo tipo, la riga del file in cui la variabile è dichiarata).

Quando il programma C++2Java incontra la definizione di un simbolo nel programma S, guarda nella tabella dei simboli per vedere se tale simbolo è già stato definito.

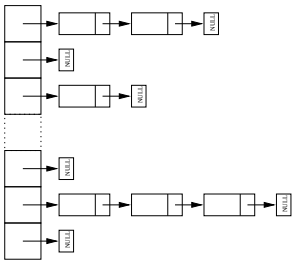
- Nel caso il simobolo sia stato già definito, mostra all'utente un messaggio di errore per comunicare che si sta definendo un simbolo già definito in precedenza.
- Se il simbolo non è ridefinito, allora il nuovo simbolo viene inserito nella tabella dei simboli.

Inoltre, può risultare necessario rimuovere un simbolo dalla tabella dei simboli, perché tale simbolo non è più necessario (e.g. una variabile è dichiarata all'interno di un blocco, quando entro nel blocco inserisco il nuovo simbolo, quando esco lo rimuovo).

Risulta quindi ovvio che su una tabella dei simboli devono poter essere effettuate le operazioni `ST_Insert`, `ST_Search`, `ST_Delete`.

Il Problema

Per poter implementare la tabella dei simboli in modo efficiente, la ditta ha deciso di realizzarla mediante una "hash table" costituita da un array `ST` di dimensione `DIMARRAY`. Gli elementi di `ST` sono delle liste (`InfoList`) di informazioni (`Info`) associate ad ogni simbolo. Il nome del simbolo è usato da una funzione `ST_Hash` per determinare un numero i compreso tra 0 e DIMARRAY-1. Nella posizione i-esima dell'array sono memorizzati in una lista (`InfoList` tutti i simboli per i quali la funzione `ST_Hash` restituisce lo stesso numero i.



Le informazioni associata ad ogni simbolo sono: il nome del simbolo (una stringa lunga al massimo 100 caratteri), il suo tipo (una stringa lunga al massimo 50 caratteri) e il numero della riga in cui il simbolo è definito (un intero). Le lunghezze delle liste si intendono compreso il terminatore (`'\0'`).

Allo studente è richiesto di implementare la struttura dati e le funzioni che consentano di realizzare una tabella dei simboli.

Quesiti

1. **(Punti: 3)** Si implementino le definizioni di tipo che servono a rappresentare i seguenti tipi:
   (a) tipo `Info` con i seguenti campi:
   `name` un array di 100 caratteri.
   `type` un array di 50 caratteri.
   `line` un intero.
   (b) tipo `InfoList`, una lista concatenata di elementi di tipo `Info`;
   (c) Un tipo `ST` avente come campi:
   array un array di 10 elementi di tipo puntatore a `InfoList`;

size un intero contenente la dimensione dell'array. Questo campo dovrà essere inizializzato al valore 10, e non dovrà essere modificato in alcun modo dal programma, e dovrà essere utilizzato come limite superiore per scorrere l'array);
Nota: allo studente è richiesto anche l'implementazione del costruttore di default `ST()` che inizializza ogni elemento dell'array a `NULL` e il campo `size` al valore 10;

Suggerimento: si utilizzino array di dimensione fissa per i campi della struttura `Info`. Non si definiscano costruttori per le strutture `Info` e `InfoList`, mentre si definisca solo il costruttore richiesto per la struttura `ST`.

2. La funzione `ST_Print` corrispondente alla seguente dichiarazione:

```
void ST_Print(ST * s)
```

che data una tabella dei simboli `s` di tipo puntatore a `ST` stampa a video il contenuto di ogni elemento dell'array in `s`, nel formato sotto riportato per una variabile `s` memorizzata in posizione 1, di tipo `Symbol Table`, definita alla riga 3.

```
1: st : Symbol Table : 3
5: i : intero : 1
```

3. **(Punti: 4)** La funzione `ST_Search` corrispondente alla seguente dichiarazione:

```
Info * ST_Search(ST * s, char key[100])
```

che data una tabella dei simboli, `s` di tipo puntatore a `ST` e un array di 100 caratteri `key`, ritorna un puntatore all'informazione associata al simbolo rappresentato da `key` se nella lista nella posizione ritornata da `ST_Hash(s, key)` occorre un simbolo con lo stesso nome, `NULL` altrimenti.

Suggerimento: si utilizzi la funzione `Compara_stringhe` per comparare due simboli. Questa funzione è così dichiarata: `bool Compara_stringhe(const char * s1, const char * s2);`

4. **(Punti: 5)** La funzione `ST_Insert` corrispondente alla seguente dichiarazione:

```
bool ST_Insert(ST * s, Info o)
```

che data una tabella dei simboli `s` di tipo puntatore a `ST`, un oggetto `o` di tipo `Info`,

- controlla se il simbolo corrispondente è già presente in `s`, ed in tal caso ritorna `false`;
- se il simbolo non è presente, allora lo inserisce nella posizione ritornata da `ST_Hash(s, key)`, essendo `key` il nome del simbolo memorizzato in `o` e ritorna `true`.

Suggerimenti: si può utilizzare la funzione `ST_Search` per verificare se il simbolo è già presente nella tabella dei simboli. Nell'inserimento di un nuovo simbolo, il candidato può decidere se inserire l'elemento in testa o in coda nella corrispondente lista.

5. **(Punti: 6)** La funzione `ST_Delete` corrispondente alla seguente dichiarazione:

```
bool ST_Delete(ST * s, char key[100])
```

che data una tabella dei simboli, `s` di tipo puntatore a `ST`, un array di 100 caratteri `key`:

- controlla se il simbolo corrispondente è presente in `s`, ed in tal caso rimuove il simbolo da `s` (deallocando la memoria non più necessaria) e ritorna `true`;
- altrimenti non fa nulla e ritorna `false`.

Suggerimenti: si può utilizzare la funzione `ST_Search` per verificare se il simbolo è presente nella tabella dei simboli. Il candidato si ricordi di gestire la cancellazione del primo elemento della lista, e la deallocazione della memoria non più necessaria.

Note importanti

1. Nella directory di lavoro `~/Esame` vi è il file: `main.cpp` che il candidato dovrà completare nelle sue parti mancanti e con i propri dati personali, e il file `st.dat` che lo studente non dovrà modificare.
2. Al fine della valutazione verrà preso in considerazione unicamente il file `~/Esame/main.cpp`. Ogni altro file verrà ignorato. Il candidato dovrà perciò assicurarsi di modificare e salvare unicamente tale file.
3. Le implementazioni delle funzioni che operano su una lista, dovranno tener conto del fatto che le liste potrebbero essere vuote.
4. Il file `main.cpp` contiene alcune funzioni di utilità che, come tali, potrebbe essere utile utilizzare.
5. Si leggano anche i commmenti e gli eventuali suggerimenti nel codice fornito nel file `main.cpp`.