APPELLO SCRITTO DI PROGRAMMAZIONE 1
CORSO DI LAUREA IN MATEMATICA
UNIVERSITÀ DEGLI STUDI DI MILANO

2018–2019

8.VII.2019

VINCENZO MARRA

Indice

Avvertenza. Il tema d'esame richiede lo sviluppo di un singolo programma che implementi alcuni semplici calcoli riguardanti insiemi i cui elementi sono caratteri. Svilupperete delle funzioni ausiliarie nel risolvere gli esercizi. Integrerete poi tutte le funzioni ausiliarie in un unico programma finale, che sarà costituito da un solo file sorgente. (Il tema d'esame richiede esplicitamente l'implementazione di alcune funzioni ausiliarie. È comunque sempre ammissibile aggiungere altre funzioni ausiliarie qualora lo riteniate opportuno per una migliore implementazione della vostra soluzione.) Si raccomanda di leggere interamente il tema d'esame con

Ultima revisione: 8 luglio 2019.

attenzione, prima di cominciare a scrivere le soluzioni dei singoli esercizi, in modo da avere chiaro l'obiettivo finale.

Principali argomenti del corso coinvolti dal tema.

- Strutture.
- Programmazione strutturata e implementazione di semplici algoritmi.
- Ricorsione.
- Allocazione dinamica della memoria.
- Passaggio degli argomenti dalla riga di comando.

Esercizio 1

Strutture dati e visualizzazione.

Punti: 3.

Definite una struttura che possa rappresentare un insieme di caratteri, e rinominatela tramite `typedef` col nome `Ins`. La struttura dovrà contenere due campi. Il primo è un puntatore a carattere di nome `e` (per 'elementi'). Il secondo è un intero di nome `n`, che rappresenta la cardinalità dell'insieme. Nell'uso delle variabili di tipo `Ins` rispetterete le seguenti convenzioni.

- Il puntatore `e` punta agli elementi dell'insieme allocati in memoria, i quali saranno dunque caratteri. Poiché gli insiemi non contengono ripetizioni, bisognerà garantire che i caratteri puntati da `e` siano sempre tutti distinti.
- Il valore di `n` è sempre un valore non negativo coincidente con la cardinalità dell'insieme rappresentato dalla variabile, e dunque col numero dei caratteri puntati da `e`.
- Il puntatore `e` è `NULL` se e solo se l'insieme rappresentato dalla variabile è vuoto. In questo caso, e solo in questo caso, `n` vale 0.

Scrivete una funzione di prototipo

```
void stampa(Ins *A)
```

che produca in uscita su `stdout` una rappresentazione dell'insieme puntato da `A`. La funzione `stampa` assume che `A` punti ad un insieme che rispetta le convenzioni riportate sopra. Per esempio, non sarà necessario gestire situazioni quali quella in cui `A` sia `NULL`. Quanto appena detto si applica a tutte le funzioni che svilupperete nel corso della prova.

La funzione `stampa` produce in uscita l'insieme rappresentato come elenco degli elementi separati da virgole, racchiuso fra graffe. Per esempio, se gli elementi di `*A` sono allocati in memoria come

a	C	b	s

la funzione `stampa` produrrà in uscita

{a, C, b, s}

(sempre senza ritorno a capo dopo la parentesi graffa chiusa). Se l'insieme puntato da `A` è vuoto, la funzione `stampa` produrrà in uscita

{}

Assicuratevi di testare il buon funzionamento della funzione `stampa` prima di proseguire. Fate analogamente per ciascuno degli esercizi che seguono.

Esercizio 2

Relazioni di appartenenza e sottoinsieme.

Punti: 6.

Implementate due funzioni di prototipo:

- `int app(char c, Ins *A)`
- `sub(Ins *A, Ins *B)`
- `int eq(Ins *A, Ins *B)`

secondo le specifiche seguenti. La prima funzione restituisce 1 se `c` appartiene all'insieme puntato da `A`, e 0 altrimenti. La seconda funzione restituisce 1 se l'insieme puntato da `A` è sottoinsieme dell'insieme puntato da `B`, e 0 altrimenti. La terza funzione restituisce 1 se l'insieme puntato da `A` è uguale all'insieme puntato da `B`, e 0 altrimenti. Implementate la funzione `sub` secondo un algoritmo ricorsivo, sfruttando anche la funzione `app`.

Esercizio 3

Funzioni per l'allocazione di insiemi.

Punti: 9.

Implementate due funzioni di prototipo:

- `int add(char c, Ins *A)`
- `Ins *strtoins(char *s)`

secondo le specifiche seguenti.

La funzione `add` modifica l'insieme puntato da `A` di modo che, dopo l'esecuzione della funzione, esso contenga il carattere `c` in aggiunta ai caratteri che già conteneva in precedenza. Si ricordi che al termine dell'esecuzione l'insieme puntato da `A` non deve contenere ripetizioni, come sempre. La funzione `add` restituisce 1 se l'operazione è andata a buon fine (nel senso che non si è riscontrato un errore di allocazione della memoria), e 0 altrimenti. Potete assumere che gli elementi dell'insieme puntato da `A` siano stati in precedenza allocati dinamicamente, di modo che eventuali chiamate a `realloc` della funzione `add` non generino errori in esecuzione.

La funzione `Ins *strtoins(char *s)` restituisce un puntatore all'insieme i cui elementi sono una copia dei caratteri che compongono la stringa `s` ricevuta in ingresso. Potete assumere che `s` non sia `NULL`. Notate che `s` può ben contenere caratteri ripetuti, mentre, al solito, l'insieme non deve contenere ripetizioni. La funzione restituisce `NULL` solo nel caso in cui si verifichino errori nell'allocazione della memoria.

Esercizio 4

Duplicazione di un insieme.

Punti: 3.

Scrivete una funzione di prototipo

$$\texttt{Ins} * \texttt{dup}(\texttt{Ins} * \texttt{A})$$

che restituisca un puntatore a una copia dell'insieme puntato da `A`, oppure `NULL` nel caso in cui durante la duplicazione si verifichi un errore nell'allocazione della memoria.

Esercizio 5

Intersezione e unione di due insiemi.

Punti: 5.

Implementate due funzioni di prototipo:

- `Ins *uni(Ins *A, Ins *B)`
- `Ins *inter(Ins *A, Ins *B)`

secondo le specifiche seguenti.

La funzione `uni` restituisce un puntatore all'insieme che è l'unione degli insiemi puntati da `A` e da `B`. La funzione restituisce `NULL` nel caso in cui durante la sua esecuzione si verifichi un errore nell'allocazione della memoria. Sfruttate la funzione `dup` dell'Esercizio 4 per l'implementazione, assieme alle funzioni sviluppate in precedenza.

La funzione `inter` restituisce un puntatore all'insieme che è l'intersezione degli insiemi puntati da `A` e da `B`. La funzione restituisce `NULL` nel caso in cui durante la sua esecuzione si verifichi un errore nell'allocazione della memoria. Per l'implementazione, sfruttate le funzioni sviluppate negli esercizi precedenti.

Esercizio 6

Procedura `main` e programma finale.

Punti: 4.

Scrivete una procedura `main` secondo le specifiche seguenti. Il programma riceve tre parametri dalla riga di comando, di cui il primo è una serie di opzioni precedute da `-`, mentre il secondo e terzo sono due stringhe (eventualmente con caratteri ripetuti) dalle quali il programma costruirà due insiemi come specificato nel seguito. Se non vi sono almeno tre parametri sulla riga di comando il programma termina con un messaggio d'errore. Se ve n'è di più, il programma considera solo i primi tre.

Il primo parametro deve cominciare col carattere `-`, a pena della terminazione del programma con un messaggio d'errore. Il parametro, dopo il carattere `-`, sarà costituito da una serie di caratteri da interpretarsi nel modo seguente.

- `I` oppure `i`. Indica al programma di computare l'intersezione dei due insiemi specificati sulla riga di comando.
- `U` oppure `u`. Indica al programma di computare l'unione dei due insiemi specificati sulla riga di comando.
- `S` oppure `s`. Indica al programma di stabilire se il primo insieme specificato sulla riga di comando sia sottoinsieme del secondo.

```
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out -iUs= "Babbo Natale" Cenerentola
{B,a,b,o, ,N,t,l,e}i{C,e,n,r,t,o,l,a}={a,o,t,l,e}
{B,a,b,o, ,N,t,l,e}U{C,e,n,r,t,o,l,a}={B,a,b,o, ,N,t,l,e,C,n,r}
{B,a,b,o, ,N,t,l,e}s{C,e,n,r,t,o,l,a}:False.
{B,a,b,o, ,N,t,l,e}={C,e,n,r,t,o,l,a}:False.
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out -iUs= "" Cenerentola
{}i{C,e,n,r,t,o,l,a}={}
{}U{C,e,n,r,t,o,l,a}={C,e,n,r,t,o,l,a}
{}s{C,e,n,r,t,o,l,a}:True.
{}={C,e,n,r,t,o,l,a}:False.
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out -UIS== abcde eacbd
{a,b,c,d,e}U{e,a,c,b,d}={a,b,c,d,e}
{a,b,c,d,e}I{e,a,c,b,d}={a,b,c,d,e}
{a,b,c,d,e}S{e,a,c,b,d}:True.
{a,b,c,d,e}={e,a,c,b,d}:True.
{a,b,c,d,e}={e,a,c,b,d}:True.
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out -=Siu "" ""
{}={}:True.
{}S{}:True.
{}i{}={}
{}u{}={}
```

FIGURA 1. Primo esempio di esecuzione del programma finale.

```
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out - a A
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out a A
Numero degli argomenti insufficiente.
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out a A B
Formato degli argomenti errato.
Vincenzos-MacBook-Pro:Soluzione enzo$ ./a.out -u a A Nel mezzo del cammin
{a}u{A}={a,A}
```

FIGURA 2. Secondo esempio di esecuzione del programma finale.

- `=` oppure `E`. Indica al programma di stabilire se il primo insieme specificato sulla riga di comando è uguale al secondo.
- Ogni altro carattere contenuto nel primo parametro è ignorato dal programma, ossia la sua presenza non ha effetto sull'esecuzione.

Il programma analizza i caratteri presenti nel primo parametro ed esegue le operazioni richieste, visualizzando i risultati nell'ordine su `stdout`. Fatto ciò, il programma termina l'esecuzione. Prestate attenzione al fatto che non vi sia memoria dinamicamente allocata al momento della terminazione del programma.

Le Figg. 1 e 2 contengono alcuni esempi di esecuzione.

Istruzioni per la consegna

- Funzioni da consegnare (in un unico file sorgente).

  `stampa`
  `app`
  `sub`
  `eq`
  `add`
  `strtoins`
  `dup`
  `uni`
  `inter`
  `main`
  Tutte le eventuali altre funzioni ausiliarie che abbiate scritto.

- Sito per la consegna:

  `https://upload.mat.unimi.it`

- Autenticarsi con le proprie credenziali di posta elettronica d'ateneo.
- Consegnare solo il codice sorgente (file `*.c`), non l'eseguibile.

(V. Marra) Dipartimento di Matematica *Federigo Enriques*, Università degli Studi di Milano, via Cesare Saldini, 50, I-20133 Milano
E-mail address: vincenzo.marra@unimi.it