UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dispense del Corso di Laboratorio di Fondamenti di Informatica II e Lab

Federico Bolelli, Silvia Cascianelli

Esercitazione 08: Liste

Ultimo aggiornamento: 20/05/2020

Attenzione!

- Prima di svolgere gli esercizi che vi proponiamo in questa videolezione vi suggeriamo di prendere visione della video soluzione dell'esercizio Load della scorsa esercitazione.
- In questo video vi spieghiamo perché le funzioni:

```
ElemType ReadElem(FILE *f);
ElemType ReadStdinElem();
```

- da questa esercitazione in avanti (anche per l'esame) avranno la seguente dichiarazione:

```
int ReadElem(FILE *f, ElemType *e);
int ReadStdinElem(ElemType *e);
```

Rubrica

- Siano date le seguenti definizioni:

```
struct Address {
        char name[40];
        char street[50];
        int  number;
        char city[30];
        char province[3];
        char postal_code[6];
};
typedef struct Address ElemType;

struct Item
{
        ElemType value;
        struct Item *next;
};
typedef struct Item Item;
```

Rubrica

- Modificate le seguenti funzioni perché possano funzionare con `ElemType` di tipo `Address`:

```
- int ElemCompare(const ElemType *e1, const ElemType *e2);
- ElemType ElemCopy(const ElemType *e);
- void ElemDelete(ElemType *e);
- int ReadElem(FILE *f, ElemType *e);
- int ReadStdinElem(ElemType *e);
- void WriteElem(const ElemType *e, FILE *f);
- void WriteStdoutElem(const ElemType *e);
```

- Basate la funzione `ElemCompare` sul campo `name` dell'indirizzo. Utilizzare a tale scopo la funzione `strcmp`.

Rubrica

- Quindi potete utilizzare le implementazioni delle primitive che vi sono state fornite per gli `ElemType` di tipo `int` (`https://github.com/prittt/Fondamenti-II`):

```
- Item* CreateEmptyList(void);
- Item* InsertHeadList(const ElemType *e, Item* i);
- bool IsEmptyList(const Item *i);
- const ElemType* GetHeadValueList(const Item *i);
- Item* GetTailList(const Item* i);
- Item* InsertBackList(Item* i, const ElemType *e);
- void DeleteList(Item* item);
- void WriteList(const Item *i, FILE *f);
- void WriteStdoutList(const Item *i);
```

Rubrica

Usando le primitive, nel file `address_book.c` si implementi la definizione delle seguenti funzioni:

Es 1) `const ElemType* Find(const Item* i, const char *name)`

La funzione prende in input una lista di indirizzi (anche vuota) e un nome. La funzione deve cercare il nome tra gli indirizzi della lista e ritornare il puntatore all'elemento corrispondente o NULL in caso il nome non sia presente.

Es 2) `Item* Delete(Item* i, const char *name)`

La funzione prende in input una lista di indirizzi (anche vuota) e un nome. La funzione deve cercare il nome tra gli indirizzi della lista ed eliminare l'elemento corrispondente (se presente) e restituire la lista risultante (puntatore alla testa).

Rubrica

Es 3) `Item* Sort(Item* i)`

La funzione prende in input una lista di indirizzi (anche vuota) e la ordina per nome usando la funzione `strcmp`. La funzione deve ritornare la lista ordinata (puntatore alla testa).

Es 4) `Item* Filtra(Item* i, const char *city)`

La funzione prende in input una lista di indirizzi (anche vuota) e una città e costruisce una **nuova lista** contenente tutti e soli gli indirizzi di quella città. Se non ce ne sono ritorna NULL.

Rubrica

Es 5) `Item* Reverse(const Item* l)`

La funzione prende in input una lista di indirizzi (anche vuota) e ritorna una **nuova lista** contenente gli stessi indirizzi, ma in ordine inverso. La lista originale non deve essere modificata.

Es 6) `Item* Append(const Item* l1, const Item* l2)`

La funzione prende in input due liste di indirizzi (anche vuote) e ritorna una **nuova lista** contenente tutti gli indirizzi della prima seguiti da tutti quelli della seconda (puntatore alla testa). Le liste originali NON devono essere modificate.

Es 7) `Item* AppendMod(Item* l1, Item* l2)`

La funzione prende in input due liste di indirizzi (anche vuote), concatena la seconda lista alla prima e ritorna l'indirizzo del primo elemento della lista risultante.

Modalità di Consegna

- Dovendo ridefinire alcune delle funzioni ausiliarie alle primitive, per questa esercitazione non sarà possibile utilizzare il sistema di sottomissione online.
- Dovrete quindi consegnare tutti gli esercizi inviando una mail a massimiliano.corsini@unimore.it , federico.bolelli@unimore.it e siliva.cascianelli@unimore.it
- **Utilizzate solo l'account UNIMORE, ogni altra mail verrà ignorata.**
- **Specificate come oggetto [fdiii]. Non fdiii o [fdi II] o altre fantasiose soluzioni**.
- Ci raccomandiamo di inviare la mail a tutti gli indirizzi!
- Bisognerà consegnare tre file: `primitive.h`, `primitive.c` e `address_book.c`. Il primo dovrà contenere le dichiarazioni dei tipi `Address`, `ElemType`, `Item`, ecc, e delle funzioni primitive e non. Il secondo dovrà contenere le implementazione delle funzioni primitive e non. Il terzo, infine, dovrà contenere le soluzioni degli esercizi proposti.
- Ricordate che la consegna è facoltativa e che avete tempo fino a venerdì 22/05/2020 compreso per inviare le soluzioni: **non inviate mail con le soluzioni nei giorni successivi perché saranno ignorate!**