Laboratorio di Programmazione 2 - Progetto 2006/2007 – Modulo 2

Il secondo modulo del progetto consiste nella definizione di funzioni per:

- la gestione di liste
- il trattamento dei nodi di documenti xsML

Tali funzioni si occupano del trattamento di strutture dati (liste e nodi) che serviranno all'implementazione dell'analizzatore sintattico (modulo 3). Nel modulo 3, infatti, gli studenti utilizzeranno queste funzioni per creare l'albero del documento xsML a seguito dell'analisi sintattica.

Funzioni per la gestione delle liste

Si richiede lo sviluppo delle seguenti funzioni:

int **new_list**(list *);

Crea una lista vuota. Restituisce OP_OK se l'operazione è andata a buon fine, oppure un errore in op_errors (vedere la specifica sotto).

int **add_first**(list *l, void *el);

Aggiunge l'elemento el in testa alla lista l. Restituisce OP_OK se l'operazione è andata a buon fine, oppure un errore in op_errors.

int **add_last**(list *l, void *el);

Aggiunge l'elemento el in coda alla lista l. Restituisce OP_OK se l'operazione è andata a buon fine, oppure un errore in op_errors.

int **is_empty**(list l);

Restituisce 1 se la lista è vuota, 0 altrimenti.

iterator **get_iterator**(list l);

Restituisce un nuovo iteratore della lista l. L'iteratore è una struttura che tiene traccia della posizione attuale all'interno della lista. L'iteratore restituito dalla funzione si trova inizialmente davanti al primo elemento.

void ***next**(iterator i);

Restituisce il prossimo elemento puntato dall'iteratore (*la successiva chiamata a next deve sempre restituire l'elemento seguente all'ultimo elemento restituito*).

int **has_next**(iterator i);

Restituisce 1 se l'iteratore non è ancora arrivato alla fine della lista (ovvero se punta a un elemento valido), 0 altrimenti.

Lo studente deve anche definire i tipi di dato **list** e **iterator**. Sarà oggetto di valutazione l'efficienza delle funzioni **add_first** e **add_last** (l'ideale sarebbe che venissero eseguite in tempo costante). Si noti che non può essere assunta una dimensione massima per il tipo di dato astratto **list**.

Funzioni di gestione di un nodo xsML

Definiamo un nodo xsML mediante la seguente struttura:

```
struct node
{
        /* nome del nodo (se e' un tag) oppure testo se e' un nodo di tipo TEXT_NODE */
        char *name;

        /* attributi del nodo */
        list attributes;

        node_type type;

        /* figli del nodo */
        list children;
};

/* il tipo node e' un puntatore a una struttura node */
typedef  struct node *node;
```

I tipi di nodo (node_type) sono definiti mediante un'enumerazione:

```
typedef enum
{
        TEXT_NODE,
        TAG_NODE
} node_type;
```

Devono essere definite le seguenti funzioni:

int **new_text_node**(node *n, char *text);
int **new_tag_node**(node *n, char *name);

Crea un nodo di tipo testo o un nodo di tipo tag impostando i campi appropriati della struttura: nel caso del nodo di tipo testo (testo passato in input alla funzione), devono essere impostati i campi name (con il testo), type (= TEXT_NODE) e children (lista vuota); nel caso del nodo di tipo tag, devono essere impostati i campi name (nome del tag passato in input alla funzione), attributes (lista vuota), type (= TAG_NODE) e children (lista vuota). La funzioni restituiscono OP_OK se vanno a buon fine o un errore in op_errors.

node_type **get_node_type**(node n);

Restituisce il tipo del nodo n.

int **add_child**(node parent, node child);

Aggiunge il nodo child alla lista dei figli (children) del nodo parent. *E' importante che sia preservato l'ordine con cui i figli sono aggiunti alla lista (il primo figlio aggiunto dovrà essere il primo elemento della lista).*

int **get_attributo**(node n, char *attribute, char **value);

Ottiene dal nodo n il valore dell'attributo attribute e lo copia nell'array di caratteri puntato da value. Restituisce OP_OK se l'operazione è andata a buon fine, oppure un codice d'errore in op_errors (*OP_NO_ATTRIBUTE_ERROR se l'attributo non è presente*). Nel caso di errore, il contenuto puntato da value non viene impostato.

int **set_attributo**(node n, char *attribute, char *value);

Imposta l'attributo attribute del nodo n con il valore value. Restituisce OP_OK se l'operazione è andata a buon fine, oppure un codice d'errore (vedere l'enumerazione op_errors). *Se l'attributo è già presente, bisogna sovrascriverne il valore. Non è importante l'ordine con cui gli attributi vengono inseriti all'interno della lista attribute.*

Codici di errore

I codici di errore sono definiti mediante l'enumerazione **op_errors**:

```
enum op_errors
{
	OP_OK = 0,
	OP_MEM_ERROR,
	OP_NO_ATTRIBUTE_ERROR,
	OP_GENERIC
};
```

Significato dei codici d'errore:

OP_OK	operazione andata a buon fine
OP_MEM_ERROR	errore di memoria
OP_NO_ATTRIBUTE_ERROR	l'attributo richiesto non è stato trovato (da utilizzare nella funzione get_attributo)
OP_GENERIC	altro errore

Consegna

Le due classi di funzioni (funzioni per le liste e funzioni per il trattamento dei nodi xsML) devono essere definite in altrettanti file .h e .c (**xsml_list**.h e .c e **xsml_node**.h e .c). In totale, quindi, dovranno essere consegnati 4 file.

L'implementazione delle liste in (ovvero, l'implementazione dei tipi di dato **list** e **iterator**) non saranno noti alle funzioni per la gestione dei nodi. Queste dovranno quindi utilizzare esclusivamente le funzioni di interfaccia definite sopra e non accedere direttamente ai campi di **list** e **iterator**.