Ferdinando Del Trionfo 566/739 Paolo Selce 50/747
Gruppo 27

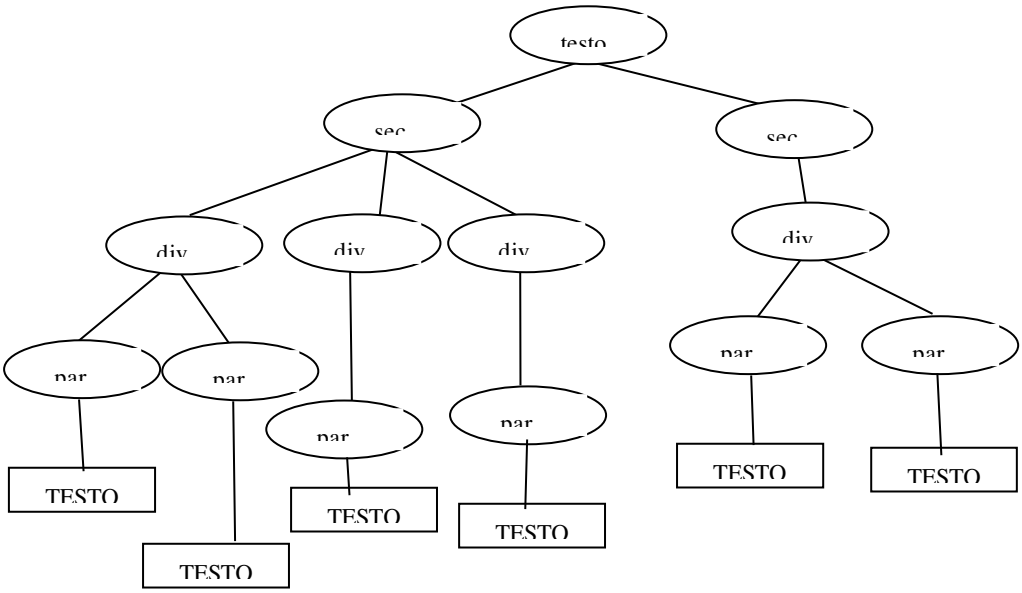
Terzo progetto

Analisi del problema:

Realizzare un software, che dato un file di testo composto da TAG e TESTO, costruisca un albero ennario. Dove ogni nodo rappresenta un TAG o il TESTO.
Acquisito l'albero ennario , trasformarlo in un albero binario.
Ogni TAG deve essere delimitato da i seguenti caratteri < > .
Ogni TAG di Apertura deve essere accoppiato da un TAG di chiusura ; il TAG di chiusura e' delimitato dai caratteri <\ >.
I TAG <bold><\bold><it><\it> fanno parte dell'insieme delle regole ma non devo essere considerati nella costruzione dell'albero.
esempio:

```
<testo>
        <sec>
                <div>
                        <par>
                                Stringa1 (con eventuali tag bold e it da ignorare)
                        <\par>
                        <par>
                                Stringa2 (con eventuali tag bold e it da ignorare)
                        <\par>
                <\div>
                <div>
                        <par>
                                Stringa3 (con eventuali tag bold e it da ignorare)
                        <\par>
                <\div>
                <div>
                        <par>
                                Stringa4 (con eventuali tag bold e it da ignorare)
                        <\par>
                <\div>
        <\sec>
        <sec>
                <div>
                         <par>
                                Stringa5 (con eventuali tag bold e it da ignorare)
                        <\par>
                        <par>
                                Stringa6 (con eventuali tag bold e it da ignorare)
                        <\par>
                <\div>
        <\sec>
<\testo>
```

diventa:


testo
sec
sec
div
div
div
div
par
par
par
par
par
par
TESTO
TESTO
TESTO
TESTO
TESTO
TESTO

dettagli funzionali:
L' utente dovra' inserire il nome del file di testo da trasformare in albero ennario, potra' scegliere la trasformazione in albero binario e la stampa dell'albero a video. L'albero verra' creato solo se il testo e' sintatticamente corretto.

note progettuali:
Visto che le strutture dati principali vengono utilizzate spesso per ogni progetto, si e' pensato di implementarle in maniera piu astratta possibile, rappresentando il campo informazione di ogni struttura dati con un puntatore a void. Tale scelta ci permette di utilizzare le stesse strutture dati con le relative funzioni semplicemente facendo un cast al campo informazione.

Fase progettuale:

Dall'analisi effettuata si evinco 4 diversi moduli per la realizzazione del progetto:

- Costruzione e stampa menu
- Caricamento da file e Costruzione albero ennario.
- Trasformazione albero ennario in albero binario
- Stampa a video

(Riutilizzo del codice e della documentazione interna esterna del primo e secondo progetto)

Analisi e descrizione del modulo Costruzione e stampa menu

La costruzione e la stampa del menu e' la parte di codice che serve all'utente per interfacciarsi con il programma. Infatti una delle richieste esplicite fatte dal cliente e' la possibilita di scegliere una delle seguenti opzioni: Verifica testo, stampa le i-esime parole,Unisci i-esime e k-esime parole. L'interfaccia grafica dovra' essere composta da un'intestazione, una sequenza di scelte, e la domanda di scelta; come di seguito:

MENU
1 Scelta1
2 Scelta2
Fai la tua scelta:

La soluzione ottimale sarebbe implementare tale menu di scelta con la possibilita di riutilizzarlo e di fare nuove aggiunte al menu in futuro. Tale scelta ci vincola a progettare la gestione del menu con file di appoggio in modo da poterli modificare.

sequenza naturale:
- Costruisci il menu
- Stampa menu

Analisi e descrizione del modulo Caricamento da file e Costruzione albero ennario.
Questo modulo prevede la costruzione di un albero ennario; dove i nodi rappresentano un TAG o rappresentano il TESTO. La costruzione dell'albero verra' effettuata se e solo se l'analisi sintattica del testo e' corretta.

sequenza naturale:
- Carica il file
- Crea Albero Ennario

Questo modulo prevede una classificazione del contenuto del file:
- TESTO
- TAG

Ogni TAG ha le stesse caratteristiche del progetto precedente ed e' racchiuso tra i caratteri speciali <>. Mentre un TESTO e' rappresento da una stringa di caratteri (<> esclusi).
N.b. La stringa composta da soli caratteri di spaziatura non e' considerata un TESTO.
I TAG <it> <bold> fanno parte delle ragole ma devono essere bypassate durante la creazione dell'albero.

Analisi e descrizione del modulo Trasformazione albero ennario in albero binario
Questo modulo prevede la trasformazione dell'albero ennario in un albero binario.

sequenza naturale:
- Carica Albero Ennario
- Trasformazione Albero Binario

Questo modulo prevede un protocollo interno che ne indichi le regole di trasformazione dell'albero

Analisi e descrizione del modulo Stampa a video
Questo modulo prevede la rappresentazione grafica a video degli alberi

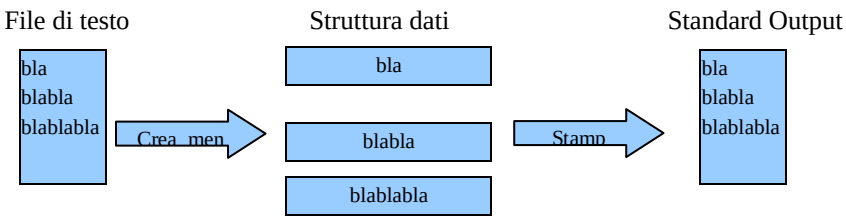
(Riutilizzo del codice e della documentazione interna esterna del primo e secondo progetto)

Costruzione e stampa del menu (Interfaccia grafica)

L'interfaccia grafica visualizzata all'utente all'avvio del programma sara' formata da un menu' composto dalle seguenti selezioni :

1 Analisi sintattica e caricamento albero ennario
2 Trasformazione albero ennario in albero binario
3 Stampa Risultati

L'idea di creare un menu riutilizzabile e aggiornabile e' la seguente: costruire una funzione che prenda da un file di testo (con una sintassi e semantica prestabilita) un Menu. Tale funzione riempira' una struttura dati contenente il menu. Tale menu potrebbe essere formattato visualizzato a proprio piacimento infatti parallelamente a questa funzione ce ne dovra essere un' altra la quale stampera' tale menu.



Pascal Like

La funzione Crea_menu(file_di_testo) prende come parametro di input una stringa che rappresenta il nome del file di testo; tale file di testo contiene la struttura del menu.
Sintassi e Semantica del file:
Il file e' composto da una parte d' intestazione, da una parte di scelte, e dall'ultima parte di domanda di selezione.
la prima riga rappresenta l'intestazione del menu.
Le righe restanti rappresentano le scelte del menu.
L'ultima riga rappresenta la domanda da visualizzare per le scelte.
valori ritornati:
La funzione ritorna un array di stringhe il quale conterra' al primo posto il nome del menu e nei restanti posti prima dell'ultimo le relative scelte; e nell'ultimo posto la domanada di scelta.
Inoltre la funzione ritorna il numero di elementi inseriti nell'array.

esempio:

File_di_testo preso in input

Menu
1-Scelta1
2-Scelta2
fai una scelta:

Array ritornato

L'array ritornato verra' stampato a video dalla funzione stampa_menu(array,elementi) che prende in input l'array ritornato dalla funzione e il numero di elementi crea_menu.

PROCEDURE e PARAMETRI
Crea_Menu(INPUT:STRING file_di_testo OUTPUT:STRING ARRAY array , INT elem)
Stampa_Menu(INPUT: STRING ARRAY array , INT elem OUTPUT:)

Dettagli implementativi (orientato a C)
Le funzioni di creazione e stampa menu' sono implementate nel file funzioni_menu.c e definite nel file funzioni_menu.h .
La funzione Crea_Menu non puo costruire menu piu lunghi di 9 scelte per evindeti motivi.
La funzione Stampa_Menu stampa a video senza formattazioni speciali.
prototipi di funzioni:
char ** Crea_Menu(char* ,int);
void Stampa_Menu(char **,int);
I dettagli dell parti implementate si possono trovare nella documentazione interna.

Caricamento da file e Costruzione albero ennario

Come su detto nell'analisi questo modulo controlla la sintassi del testo e costruisce un albero ennario dal testo preso in input.
L'obbiettivo principale e' sempre quello di riutilizzare il vecchio codice, per questo tale modulo e' stato strutturato come segue:

- Analisi sintattica del file **(Riutilizzo del codice e della documentazione interna esterna del secondo progetto)**
- Scansione del file
- Riconoscimento TAG da TESTO
- Inserimento nell'albero

Messo da parte il primo punto, l'idea di massima per creare un albero ennario avente le caratteristiche estrapolate dall'analisi, e' quella di riconoscere i TAG dal TESTO , tenere traccia delle priorita di inglobamento dei TAG e aggiungere i nodi all'albero. Fatto questa premessa le domande da porsi sono:
come tenere traccia delle priorita di inglobamento e in che modo si devono relazionare i nodi dell'albero.
Si puo pensare di mettere in uno stack i riferimenti dei nodi dell'albero creati dei relativi TAG di apertura scansiti; e poi relazionare con grado di parentela figlio il nodo creato e il nodo della vecchia testa.
Quando poi si trova poi un TAG di chiusura allora viene estratto il nodo sulla testa dello Stack.
Di seguito c'e' schema che illustra tale funzionamento:

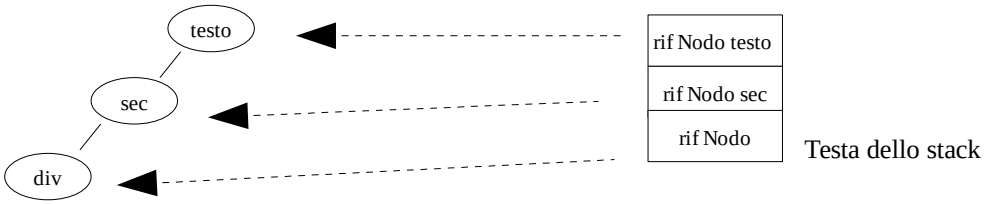
Testo preso in input

```
<testo>
    <sec>
        <div>
        <\div>
        <div>
        <\div>
     <\sec>
<\testo>
```

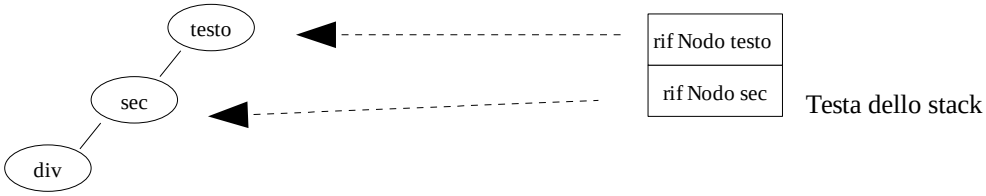
Scansione di <testo> (creo l'albero e inserisco il riferimento nello stack)

Scansione di <sec> (creo il nodo sec e inserisco il riferimento nello stack)

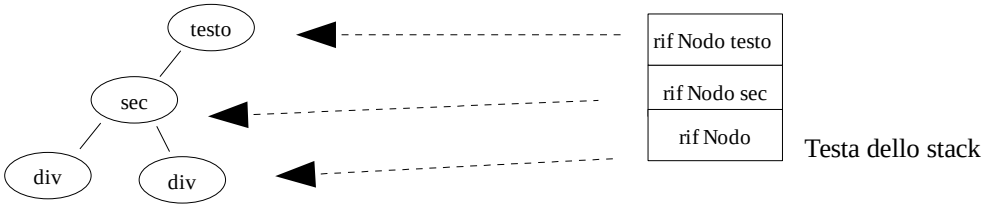
Scansione di (creo il nodo div e inserisco il riferimento nello stack)



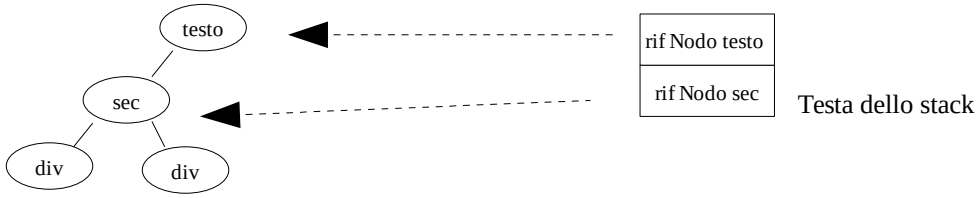
Scansione di <\div> (estraggo il riferimento nello stack)



Scansione di (creo il nodo div e inserisco il riferimento nello stack)

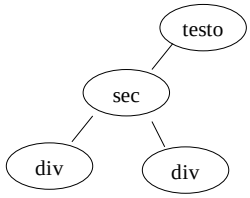


Scansione di <\div> (estraggo il riferimento nello stack)



Scansione di <\sec> (estraggo il riferimento nello stack)

Scansione di <\testo> (estraggo il riferimento nello stack)



stack vuoto

LEGGENDA
rif Nodo rappresenta il riferimento al nodo puntato dalla freccia

Sostanzialmente tale idea e molto simile a qulla del controllo sintattica del progetto precedente solo

che cambia nell'inserimento dei nodi dell'albero.

Il primo punto, Analisi sintattica del file e' gia' stata implementata nel progetto precedente.
Le funzioni che implementano questo modulo si trovano nel file funzioni_analizza_file.c e gli header nel file funzioni_analizza_file.h .
La funzione utilizzata e'

int Analizza_Testo(char* File_Di_Testo)

Questa funzione prende un path di un file di testo da analizzare in input.
Ritorna 0 se la correttezza e' verificata, 1 se c'e' un errore di TAG, 2 se c'e' un errore di stack.
L'albero viene creato se e solo se tale funzione ritorna 0.

Il secondo e terzo punto sono delle funzionalita' gia presenti nella funzione su citata Analizza_Testo. Abbiamo pensato quindi di riutilizzare tale funzione con delle piccole modifiche. Tale funzione nel progetto precedente eseguiva i seguenti passi: scansisce il testo; se e' un TAG di apertura ed e' valido, allora viene inserito in testa allo stack, se e' un TAG di chiusura viene estratto il primo elemento in testa allo stack, e se e' un TESTO, viene controllato se il TAG in testa allo stack lo supporta. La Nuova funzione quindi verra' modificata come segue:

Scansisco il testo

Se trovo un TAG di apertura:
    Se il TAG trovato e' il primo del testo:
        creo un albero
    altrimenti:
        creo un nodo di un albero e lo faccio diventare figlio del nodo sulla testa dello stack
  Inserisco nello stack il nodo creato

Se trovo un TAG di chiusura:
  Estraggo il riferimento del nodo sulla testa dello stack

Se trovo un TESTO:
  creo un nodo di un albero e lo faccio diventare figlio del nodo sulla testa dello stack

Strutture dati e routine fondamentali

Questo modulo prevede l'utilizzo degli alberi; tutte le funzioni e i dettagli implementativi sono riportati nella sezione successiva(strutture dati e routine fondamentali).
In questo modulo per ogni nodo dell'albero il campo informazione e' un puntatore a stringa, ogni stringa puo' contenere un TAG oppure un TESTO; per come sono trattate le informazioni dei nodi dell'albero bisogna fare un cast a puntatore a char.
Questa scelta viene fatta perche' cosi il campo informazione si puo distinguere facilmente. Infatti un TAG e' racchiuso tra i caratteri speciali <> mentre un testo non lo e'.

Un' altra struttura dati utilizzata e' lo stack. Esso e' stato illustrato nel dettaglio nel progetto precedente. Il campo informazione dello stack contiene un puntatore ad un nodo dell'albero.

Dettagli implementativi

Nell'analisi di questo modulo si evince un piccolo dettaglio legato su dei tag particolari essi sono <it><bold>. Infatti essi fanno parte delle regole sintattiche ma nella creazione dell'albero non devono ne comparire come nodi ne comparire nel testo.
Questo problema puo essere risolto facilmente controllando il tipo di TAG.
Considerando che i tag prima citati sono dei tag legati sostanzialmente al testo (significa che e' molto probabile che essi si trovino nel testo) e che un TESTO puo essere formato da puo TAG it e bold.
esempio:
<par> Ciao siamo il <it>gruppo <bold>27<\bold> <\it> <\par>
in questo caso il TESTO e' : Ciao siamo il gruppo 27   che dovra essere una foglia del TAG <par>.
nota per come era strutturata la vecchia funzione il while generale ciclava in modo tale da caricare o un TAG (qualunque esso sia) o un TESTO quindi e' stata fatta una modifica per caricare tutto il TESTO esclusi i TAG particolari.

quindi la sequenza di istruzioni sara':

scansisco il testo:
- se trovo un tag di apertura diverso da: <it> o <bold>
    Se esiste l'albero   creo il nodo dell'albero che diventa figlio del nodo sullo stack
                altrimenti creo l'albero
    inserisco il puntatore del nodo creato sulla testo dello stack

- se trovo un tag di chiusura diverso da: <\it> o <\bold>
    Se sulla testa dello stack trovo un TAG, faccio un pop dalla testa dello stack (caso fine tag)
    Se sulla testa dello stack trovo un TESTO, faccio due pop dalla testa dello stack (caso fine tag e fine testo perche' in testa c'e' del il TESTO)
nota: il doppio pop viene fatto perche la funzione quando trova il tag di chiusura(dell'ultimo TAG sulla testa dello stack), deve estrarre dalla testa dello stack prima il TESTO e poi il TAG.

- se trovo un tag di apertura o chiusura di tipo: <it> o <bold>
leggo la stringa successiva. Quindi essi vengono bypassati

- se trovo del TESTO
Controllo la testa dello stack:
se sulla testa c'e' un puntatore a nodo di albero nel quale campo informazione contiene un TAG
creo un nodo dell'albero con il TESTO
inserisco il nodo dell'albero contenente TESTO sulla testa dello stack
inserisco il nodo che contiene il TESTO come figlio del nodo sulla testa dello stack
se trovo un puntatore a nodo di albero nel quale campo informazione contiene TESTO
Concateno il TESTO trovato, con il TESTO contenuto nel nodo della testa dello stack
in questo modo risolvo il problema su citato cioe quello di ignorare i tag particolari

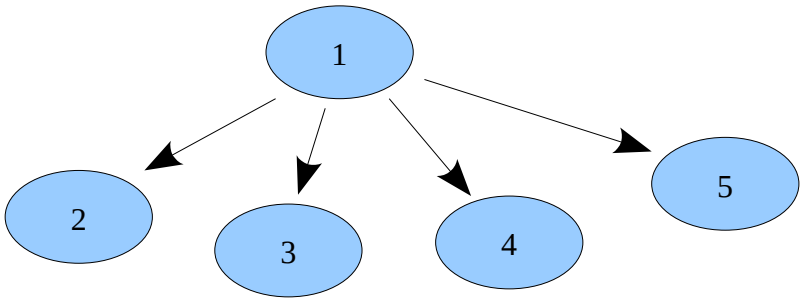
Motivazione scelte:
Si puo' pensare che scansire due volte il testo (uno per l'analisi della correttezza e l'altro per la creazione dell'albero) grava sulla velocita di esecuzione perche' viene fatto l'accesso al file 2 volte, effettivamente cosi' e' , ma questa scelta va tutto a vantaggio sulla chiarezza di lettura dell'algoritmo e del riutilizzo del codice senza grandi sforzi.

PARAMETRI E FUNZIONI

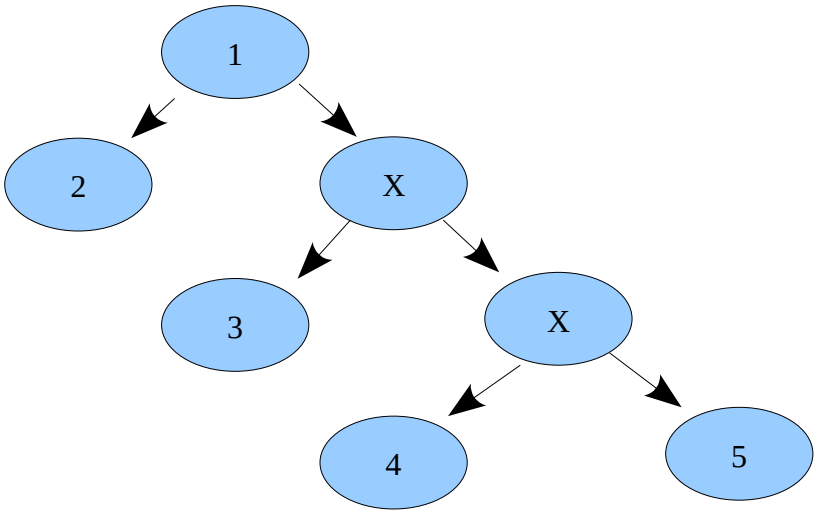
Funzione che crea e ritorna un albrero ennario
INPUT: path del file di testo
OUTPUT:ritorna la radice dell'albero se e' tutto corretto, NULL se c'e' stato qualche errore
NODO *Crea_Albero(char* File_Di_Testo)
possibili errori L'albero non viene creato se il testo non e' sintatticamente sintatticamente corretto. Eventuali errori di Stack overflow. File di testo inesistenti.
La seguente funzione e' definita nel file funzioni_creazione_albero.c e funzioni_creazione_albero.h

Trasformazione albero ennario in albero binario

Questo modulo prevede la trasformazione di albero ennario in albero binario.
Il problema principale da risolvere per questo modulo e' legato a come bisogna rappresentare piu di 2 fratelli in albero binario.
Sostanzialmente bisogna stabilire un protocollo interno per definire le regole di rappresentazione.
Una possibile idea potrebbe essere quella di creare un nodo fittizio che abbia una etichetta particolare per esempio X e che sta a significare che : il figlio di X e' figlio del padre di X, questa definizione ovviamente continua ricorsiva finche' non si trova un nodo diverso da quello fittizio.
Esempio
supponiamo di avere il seguente albero:



Quest albero rappresenta un albero ennario dove i numeri rappresentano le etichette.
secondo il ragionamento su fatto quest albero verrebbe trasformato come segue



Come si capisce dal disegno se volessi sapere chi e' il padre di 4 basta salire a ritroso affinche non si trova un avo differente dall'etichetta X

Definito come un albero ennario viene trasformato in binario bisogna definire i passi da seguire per realizzare tale sistema.
Sostanzialmente l'idea e quella di accedere ricorsivamente fino alle foglie dell'albero poi risalire a ritroso (dalla ricorsione) e trasformare i figli del nodo corrente in albero binario con il protocollo su citato.

Dettagli implementativi

La funzione principale di questo modulo e' EnnarioToBinario. Essa esegue per ogni nodo con piu di 2 figli una funzione fondamentale: ListToAlberoBin
la quale trasforma una lista di nodi di alberi in albero binario con il protocollo su citato.
In questo caso L'etichetta fittizia utilizzata e' X
Tali funzioni sono implementate nel file albero.c e definite in albero.h

PARAMETRI E FUNZIONI

Questa funzione trasforma un albero ennario in albero binario con il seguente protocollo: se esistono piu di 2 nodi fratelli allora vinene creato un nodo fittizio X il quale significa che il il padre del figlio di X e' il padre di X ovviamente procedendo ricorsivamente verso l'alto finche non si trova una etichetta diversa da X
INPUT: radice dell'albero da convertire,Etichetta fittizia X
OUTPUT: albero binario
NODO *EnnarioToBinario(NODO *Albero,void* Etichetta)

Nota: Etichetta e' di tipo *void. In questo progetto viene passato un * char, ma in qualsiasi altro progetto l'etichetta fittizia potrebbe essere una qualsiasi area di memoria.

Trasformazione albero ennario in albero binario

Tale modulo prevede la stampa a video grafica degli alberi.
la funzione che implementa tale modulo attraversa ricorsivamente l'albero e nodo che trova stampa a video il suo contenuto.
La rappresentazione grafica dell'albero viene cosi fatta:
-La stessa colonna rappresenta lo stesso grado di parentela
-il nodo successivamente sotto reppresenta il figlio
esempio

```
<testo>
   |     <sec>
   |        |      <div>
   |        |         |      <par>
   |     <sec>     |
   |        |      <div>
```

Dettagli implementativi

il parametro Tab rappresenta il livello di profondota' della funzione ricorsiva e di conseguenza anche il livello dell'albero.Tale parametro mo serve per tabulare i TAG

PARAMETRI E FUNZIONI

Stampa Ricorsivamente i campi info dell'albero
INPUT: Radice dell' albero,funzione che stampa l'informazione del nodo,livello dell'albero
OUTPUT: stampa grafica a video
void Stampa_Ric_Albero(NODO *Albero,void (*Print) (NODO *),int Tab)

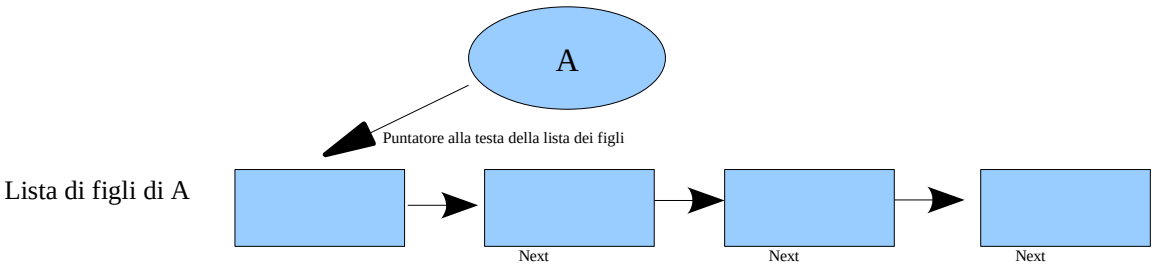
Nota: Print e' una funzione implementata a parte che stampa il contenuto del nodo di un albero

Strutture dati e Routine fonfamentali

Le strutture dati fondamentali utilizzata per questo progetto sono gli alberi.
Gli alberi sono strutture dati dinamici che possono modificare la loro dimensione a RUNTIME.
Essi sono composti da nodi che stanno in relazione con altri nodi.
Le relazioni tra nodi sono di parentela, infatti un nodo puo essere padre o figlio di un altro nodo.
Gli alberi hanno le seguneti caratteristiche:
esiste un nodo che si chiama radice. Esso e' un avo di tutti gli altri nodi e non ha un nodo padre
Ogni per ogni nodo A esiste un e un solo percorso per ogni nodo B
I nodi che non hanno figli sono chiamati foglie, mentre gli altri nodi interni.
Un albero viene definito binario se e solo se ha al piu due figli.
Ogni nodo di un albero viene rappresentato dalle seguente struttura dati:

```
typedef struct Nodo {
void *Elem;   /*Campo informazione*/
int QuantNodi;   /*Numero di figli*/
struct List *Testa;     /*Lista dei figli*/
} NODO;
```

Il puntato Elem rappresenta il puntatore dell'informazione di quel nodo.
La variabile QuantNodi rappresenta la quantita di figli contenuti nella lista.
il puntatore a struct List rappresenta il puntatore alla testa della lista dei suoi figli.
Ogni campo informazione della lista contiene il puntatore alla radice del sotto albero figlio.
esempio:



E' Stata fatta la scelta di inserire i figli in una lista in modo tale nel caso si voglia aggiungere un nuovo sotto albero ad un nodo basta aggiungere il nodo alla lista del nodo dell'albero.

Le funzioni dettagliate e documentate delle liste stanno nella sezione routine e strutture dati fondamentali del secondo progetto!

PARAMETRI E FUNZIONI

questa funzione alloca un nodo di un albero a lo inizializza
INPUT: Elem e' il puntatore all'informazione del nodo
OUTPUT: Ritorna il riferimento al nodo
NODO *CreaNodo (void *Elem)

questa funzione inserisce un nodo figlio al nodo padre. La funzione alloca un nodo e nel campo Elem del nodo assegna la variabile Elem
INPUT: Nodo padre, elemento da inserire
OUTPUT: ritorna il riferimento al nodo creato
NODO* InsNodo (NODO* Padre, void* Elem)

questa funzione inserisce un nodo in un albero binario
INPUT: Padre rappresenta il nodo padre, elem rappresenta il puntatore dell'elemento da inserire
OUTPUT: Ritorna 0 se il figlio non e' stato inserito, perche' non rispetta le regole
di albero binario, ritorna 1 invece se il figlio e' stato inserito
int InsNodoBin (NODO* Padre, void* Elem)

questa funzione inserisce un nodo in un albero ennario
INPUT: Padre rappresenta il nodo padre, elem rappresenta il puntatore dell'elemento da inserire nel campo info del nodo
OUTPUT: Ritorna il riferimento al nodo se inserito, ritorna NULL altrimenti
NODO *InsNodoEnn (NODO* Padre, void* Elem)

questa funzione crea la radice di un albero
INPUT: Elem rappresenta l'elemento da inserire nella radice
OUTPUT: ritorna la radice dell'albero
NODO*CreaAlbero (void* Elem)

questa funzione ritorna il campo informazione del nodo (puntatore Elem)
INPUT: Nodo dell'albero
OUTPUT: ritorna il puntatore all'informazione
void *GetInfo(NODO* Nodo)

Ritorna la lista dei figli
INPUT: Padre rappresenta il padre dei figli
OUTPUT: Ritorna la lista dei figli del padre
LIST *GetFigli(NODO *Padre)

Questa funzione trasforma un albero ennario in albero binario con il seguente protocollo: se esistono piu di 2 nodi fratelli allora vinene creato un nodo fittizio X il quale significa che il il padre del figlio di X e' il padre di X ovviamente procedendo ricorsivamente verso l'alto finche non si trova una etichetta diversa da X
INPUT: radice dell'albero da convertire,Etichetta fittizia X
OUTPUT: albero binario
NODO *EnnarioToBinario(NODO *Albero,void* Etichetta)

Questa funzione costruisce un albero binario a partire da una lista.L'albero viene costruito secondo il seguente protocollo
il padre di ogni figlio ha una etichetta fittizia,se la lista presa in input e nulla allora il valore di ritorno e' nullo, se la lista contiene un solo elemento allora la radice e' proprio il nodo contenuto nella lista.
INPUT: Lista che nel campo informazione=Elem contiene un nodo di un albero, puntatore all'etichetta
OUTPUT: Radice dell'albero costruito
Attenzione il contenuto di Etichetta non deve essere una varibile locale di qualche funzine

ma deve stare o nello HEAP o nella memoria STATICA
NODO *ListToAlberoBin(LIST * Nodi,void *Etichetta)

Funzioni ausiliarie

Le funzioni ausiliarie utilizzate sono:
char read_char(char * Exp)
char *read_string(char * Exp,int Dim)
entrambe sono funzioni di controllo sui caratteri. Esse vengono utilizzate per i caratteri delle scelte e per i path dei file.
read_char legge da stdin un carattere se esso e' contenuto nella stringa Exp allora viene ritornato.
read_string legge da stdin una sequenza di caratteri: alfanumerici e numeri in piu uno dei caratteri nella stringa Exp.
read_char per le scelte prende in input la seguenti stringa 123456789 perche le scelte possibili sono 9; mentre read_string prende in input la stringa ./\\ che rappresentano i caratteri essenziali per definire un path.

Funzione Principale

La funzione principale per le possibili scelte viene implementata con una struttura decisionale SWITCH CASE ripetuta in un WHILE.

Crea_Menu(INPUT: FILE file_di_menu OUTPUT: STRING Array, INTEGER elementi)
Stampa_Menu(INPUT: STRING Array, INTEGER elementi) //stampa a video il menu

Attendi un carattere di scelta da testiera
Scelta_menu=read_char(1234)

SCELTA DAL MENU:

1 Crea_Albero(INPTU:Path_File OUTPUT: NODO * Albero)
2 EnnarioToBinario(INPUT:NODO * Albero,void * Etichetta integer i OUTPUT: NODO*)
3 Stampa_Ric_Albero(INPUT:NODO * Albero, PUNTATORE A FUNZIONE,Integer OUTPUT: LIST)
3 EXIT

File Essenziali:

funzioni_analizza_file.c
funzioni_analizza_file .h
funzioni_menu.c
funzioni_menu.h
funzioni_char.c
funzioni_char.h
funzioni_carica_parole.c
funzioni_carica_parole.h
funzioni_creazione_albero.c
funzioni_creazione_albero.h
albero.c
albero.h
list.c
list.h
main.c

Menu	Contiene la struttura del menu
testo	File che contengono tasto da analizzare
testo1	File che contengono tasto da analizzare

Raccomandazioni D uso:

Il file che contiene la struttura del Menu de stare nella cartella dove viene eseguito il programma