Fondamenti di Informatica T-1

Contenuto di questa esercitazione

- Esercizi sulle grammatiche
- Esercizi sulla rappresentazione dei numeri
- Esercizi di analisi
- Esercizisui record di attivazione
- Esercizi di sintesi

Esercizio - Grammatiche (1)
Espressioni algebriche

G = ⟨VT,VN,P,S⟩, dove:

VT = { +, -, *, /, (,), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 }

VN = {<E>, <T>,
<F>, <num>,
<cifra>, <cifra-non-nulla> }

S = <E>

Esercizio - Grammatiche (1)
Espressioni Algebriche

P = {

<E> ::= <E> + <T> | <E> - <T> | <T>

<T> ::= <T> * <F> | <T> / <F> | <F>

<F> ::= <cifra> | (<E>)

<cifra> ::= 0 | <cifra-non-nulla>

<cifra-non-nulla> ::= 1|2|3|4|5|6|7|8|9

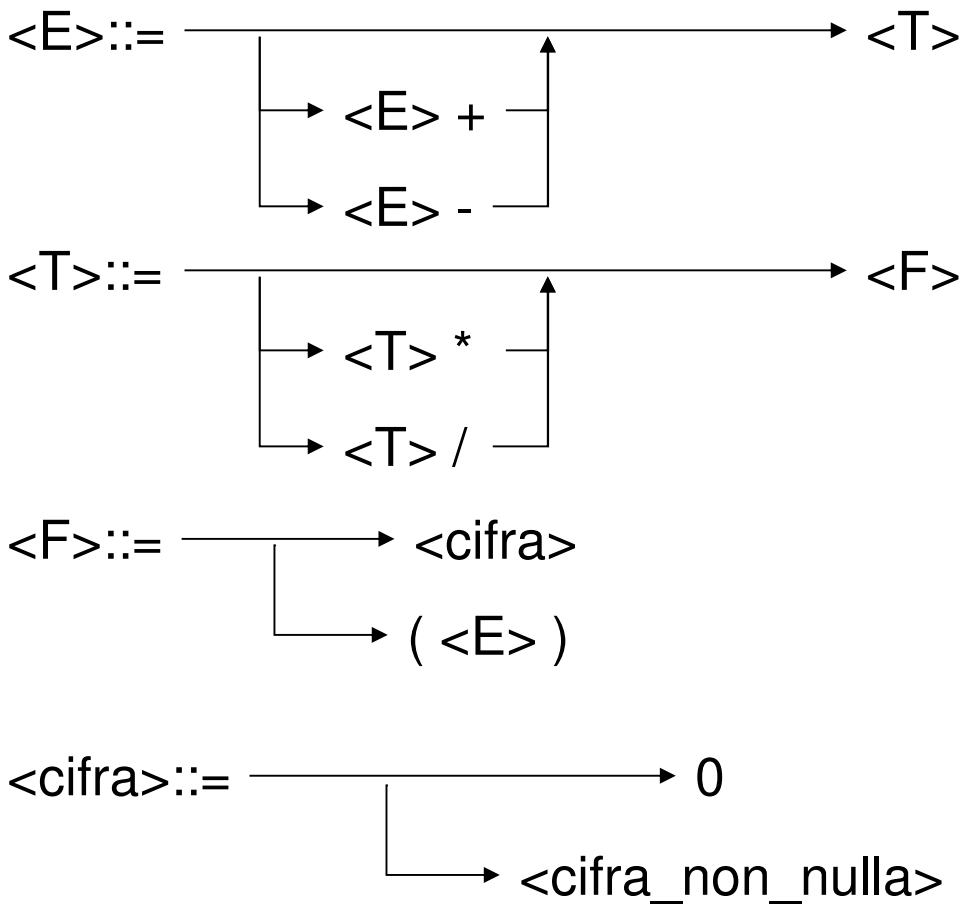
}

Disegnare il diagramma sintattico di tale grammatica. Determinare poi se le seguenti frasi fanno parte del linguaggio generato da questa grammatica o no, e disegnarne l'albero di derivazione sintattica, e la derivazione left-most:

1. **1 + 4 / 2**
2. **(1 + 4) / 2**

Esercizio - Grammatiche (1)
Soluzione

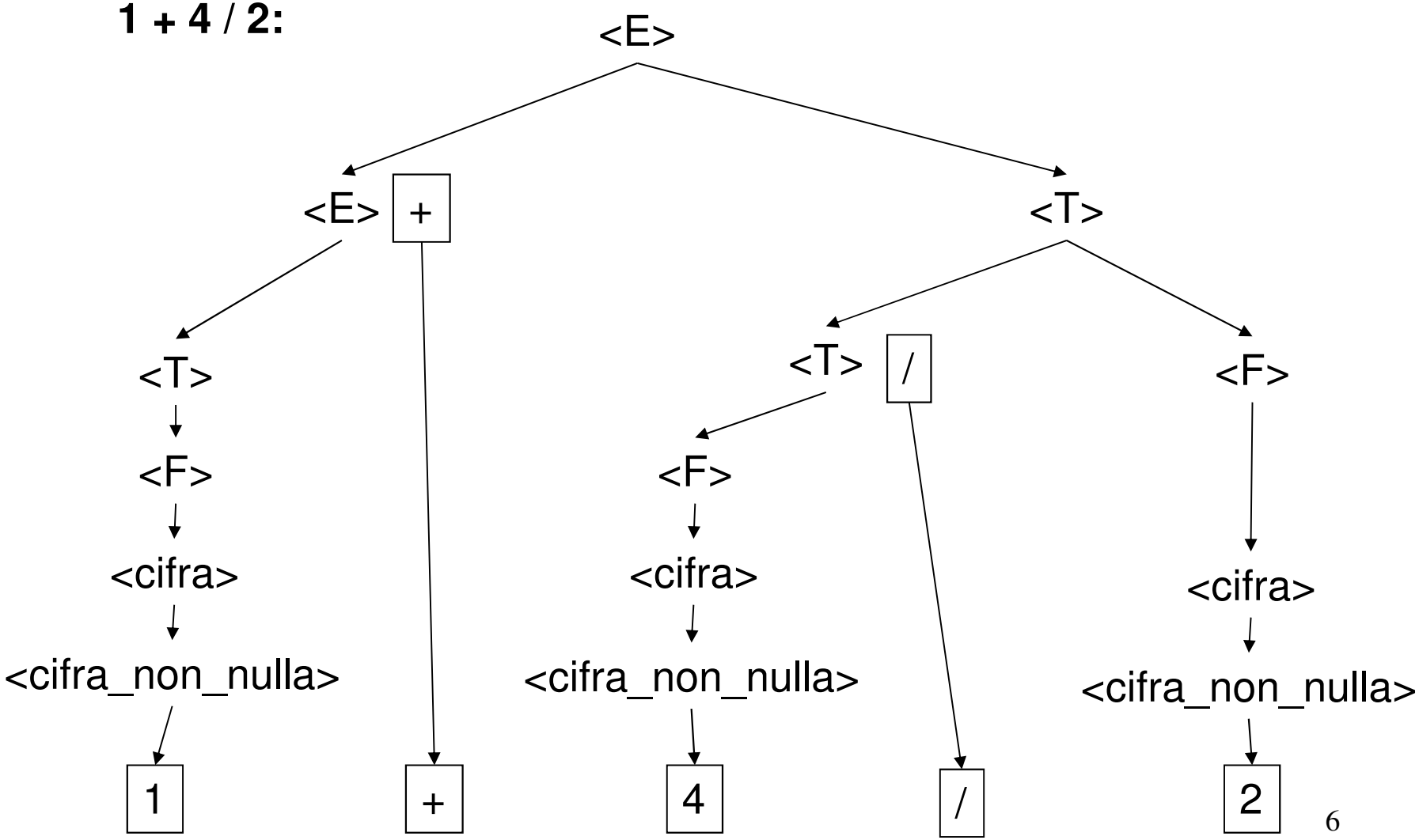
Diagramma sintattico:



Esercizio - Grammatiche (1)

Soluzione

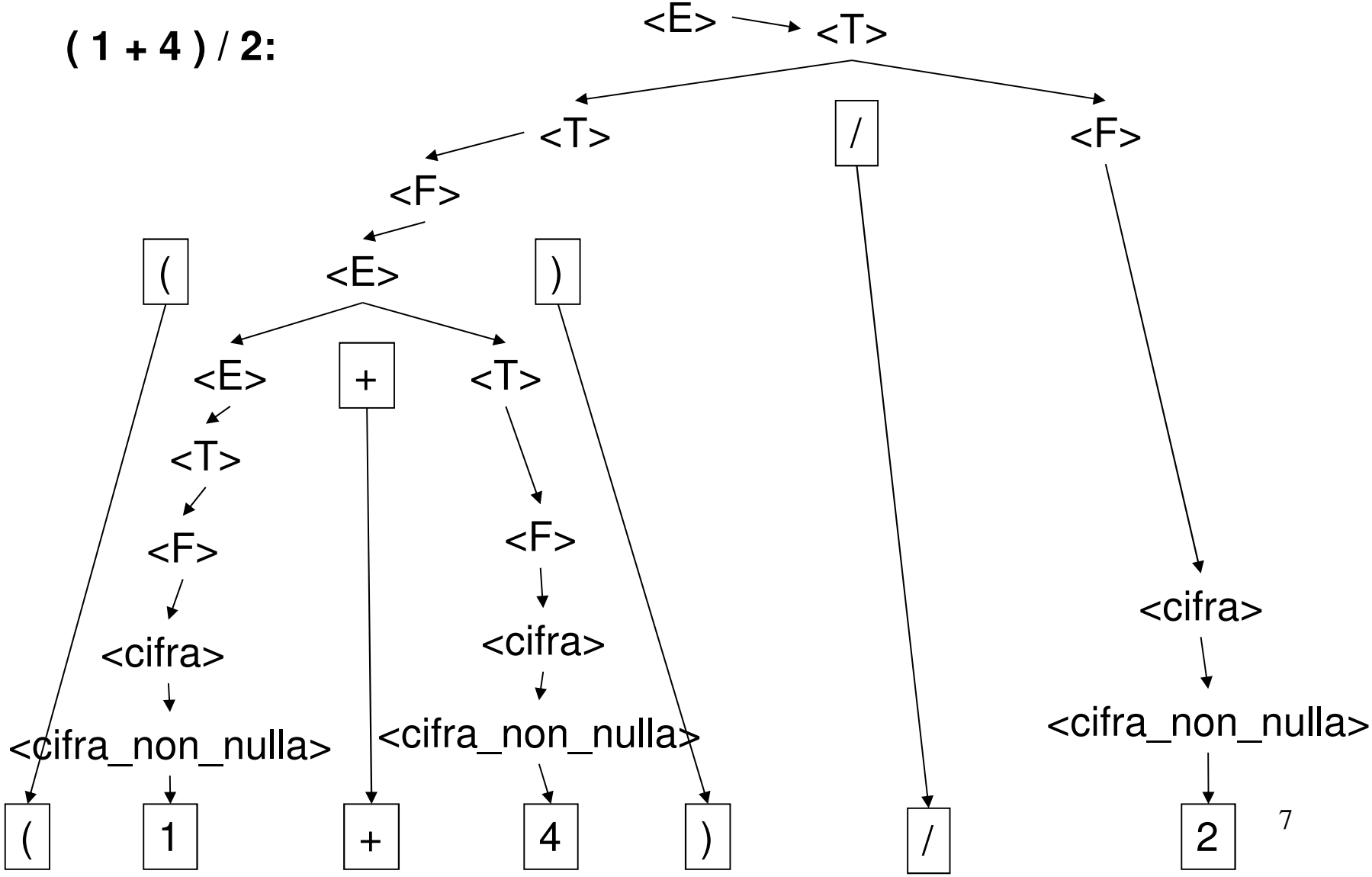
1 + 4 / 2:



Esercizio - Grammatiche (1)

Soluzione

(1 + 4) / 2:



Esercizio - Grammatiche (1)
Soluzione

Derivazione left-most:

1. 1 + 4 / 2

<E> → <E> + <T>
→ <T> + <T>
→ <F> + <T>
→ <cifra> + <T>
→ <cifra_non_nulla> + <T>
→ 1 + <T>
→ 1 + <T> / <F>
→ 1 + <F> / <F>
→ 1 + <cifra> / <F>
→ 1 + <cifra_non_nulla> / <F>
→ 1 + 4 / <F>
→ 1 + 4 / <cifra>
→ 1 + 4 / <cifra_non_nulla>
→ 1 + 4 / 2

Esercizio - Grammatiche (1)
Soluzione

Derivazione left-most:

2. (1 + 4) / 2

<E> → <T>
→ <F> / <F>
→ (<E> + <T>) / <F>
→ (<F> + <T>) / <F>
→ (<cifra_non_nulla> + <T>) / <F>
→ (1 + <F>) / <F>
→ (1 + <cifra_non_nulla>) / <F>
→ (1 + 4) / <cifra>
→ (1 + 4) / 2

→ <T> / <F>
→ (<E>) / <F>
→ (<T> + <T>) / <F>
→ (<cifra> + <T>) / <F>
→ (1 + <T>) / <F>
→ (1 + <cifra>) / <F>
→ (1 + 4) / <F>
→ (1 + 4) / <cifra_non_nulla>

Esercizio - Grammatiche (2)
Compito del 13 Settembre 2007

Si consideri la grammatica G con scopo S, simboli non terminali {X, Y, F, G} e simboli terminali {a,b,c,0,1,2}:

S ::= GXY | FX
X ::= FX | FYX
Y ::= G | GYF
F ::= 0 | 1 | 2
G ::= a | b | c

La stringa "12ac0b" appartiene al linguaggio generato da tale grammatica?

In caso affermativo, se ne mostri la derivazione left-most.

Esercizio - Grammatiche (2)
Compito del 13 Settembre 2007

La stringa data inizia col simbolo terminale '1' e termina col simbolo terminale 'b'.

Per poter ottenere una tale stringa, dovrebbe esserci una produzione con F come primo simbolo non terminale e G come ultimo simbolo non terminale…

… ma nessuna delle produzioni date ha tali caratteristiche. Infatti l'unica produzione che può terminare con G è S::= GXY, ma G come primo simbolo non terminale è incompatibile con la stringa data.

La stringa data non appartiene quindi al linguaggio generato dalla grammatica data.

Esercizio - RAPPRESENTAZIONE (3)

Un elaboratore rappresenta i numeri interi su 8 bit in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

39 + (- 91)

Esercizio - RAPPRESENTAZIONE (3)

+39 -> 00100111

91 -> 01011011

10100100

10100101 -> -91

Conversione del valore assoluto in base 2

Inversione dei bit

Aggiungo 1

Si esegue la somma:

00100111 +
10100101

11001100

11001100
00110011
00110100 -> 52 (base 10)

Complemento a 2 – quando c'è overflow?

Nella rappresentazione a complemento a 2, l'overflow può accadere solo in conseguenza di:

- Somma di due numeri positivi
- Somma di due numeri negativi

L'overflow si verifica quando uno dei due ultimi bit di riporto è a uno, ma non lo sono entrambi (xor degli ultimi due bit di riporto)

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri positivi senza overflow (caso su 4 bit):

```
(riporto)      0 0 0 0
3       ->     0011
4       ->     0100
               ---------
               0111
```

xor(0, 0) = 0, non c'è overflow!

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri positivi *con* overflow (caso su 4 bit):

```
(riporto)      0 1 1 1
3       ->     0011
5       ->     0101
               ---------
               1000
```

xor(0, 1) = 1, OVERFLOW!!!

Sommando due numeri positivi, è stato modificato il bit del segno…
… deve essere successo qualcosa!!!

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri negativi *senza* overflow (caso su 4 bit):

(riporto) 1 1 0 0

-3 -> 1101

-4 -> 1100

1001

xor(1, 1) = 0, no overflow

Il bit del segno è rimasto invariato…
… ed entrambi i riporti sono ad 1…

Complemento a 2 – quando c'è overflow?

Consideriamo il caso di due numeri negativi *con* overflow (caso su 4 bit):

```
(riporto)      1 0 0 0
-3      ->     1101
-6      ->     1010
              --------
              0111
```

xor(1, 0) = 1, OVERFLOW!!!

Il bit del segno è stato modificato…
…cosa è successo ?

Esercizio - RAPPRESENTAZIONE (4)

Un elaboratore rappresenta i numeri interi su 8 bit in complemento a 2. Indicare come viene svolta la seguente operazione aritmetica e determinarne il risultato traslandolo poi in decimale per la verifica:

(-13) - (-91)

Esercizio - RAPPRESENTAZIONE (4)

13 -> 00001101
11110010
11110011 -> -13

-(-91) -> 01011011

(riporti) -> 1 1 1 1 1 1
-13 -> 1 1 1 1 0 0 1 1 +
-91 -> 0 1 0 1 1 0 1 1
---
0 1 0 0 1 1 1 0

01001110 -> 78 in base 10

Esercizio analisi (5)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori di v e di N al termine dell'esecuzione? (si motivi opportunamente la risposta data)

```
int main(void)
{
     int N=4, i, v;

     v = 0;
     {  ++N; }

     do {
        if ((N%2) == 1)
                v = v + N;
        else
                v = v - N;
        N--;
     } while (v > 4);

     for (i=0; i<N; i++)
        v = v - 1;
     for (i=0; i < (i?N:0); )
        v = 10;
     return (0);
}
```

Esercizio analisi (5)

Il programma compila correttamente, e al termine dell'esecuzione le variabili `v` ed `N` valgono rispettivamente -2 e 3

- Le prime due istruzioni pongono v a 0, e poi incrementano il valore di N, che da 4 viene a valere 5.
- Il primo ciclo che poi si incontra nel codice è di tipo do..while, perciò almeno una volta viene eseguito. In particolare, l'istruzione di if controlla: se N è dispari allora v viene incrementato di N, altrimenti v viene decrementato di N. Il ciclo termina quando la condizione (v>4) non vale più. Alla prima iterazione dunque v viene a valere 5, ed N è decrementato a 4; alla seconda iterazione v viene a valere 1 ed N decrementato a 3: Siccome la condizione di continuazione del ciclo non è più verificata, il ciclo si interrompe.
- Il secondo ciclo che si incontra decrementa di 1 il valore di v, per N volte. Quindi al termine v viene a valere -2.
- Il terzo ciclo presente nel programma non viene eseguito, poichè la condizione è immediatamente non verificata.

Esercizio analisi (6)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```
#include <stdio.h>
void F(int v[], int pos) {
     int N=4;   N--;
     *(v+pos+1) = *(v+pos+1) + *(v+pos);
     return; }

int main() {
     int N=4, v[5], i;
     {  ++N; }

     for (i=0; i<N; i++)
        v[i]=2*i+1;

     for (i=0; i<N-1; i++)
        if (v[i]) F(v, i);

     printf("Adesso N vale: %d\n", N);
     for (i=1; i++ < (i?N:0); )
        printf("%d\n", v[i-1]);
     return 0; }
```

Esercizio analisi (6)
Soluzione

Il programma compila correttamente e stampa:

```
Adesso N vale: 5
4
9
16
25
```

- la funzione **F** si limita a modificare il valore dell'array con indice (**pos+1**) sommandogli il valore contenuto all'indice **pos**,
- Nel **main**, il primo ciclo **for** provvede ad inizializzare l'array **v** ai valori {**1,3,5,7,9**}; il secondo ciclo invece invoca ripetutamente la funzione **F**, ed all'uscita dal ciclo il vettore viene a valere {**1,4,9,16,25**}.
- Infine viene stampato il il valore di **N** (che in seguito al preincremento inziale passa da 4 a 5), e poi con un ultimo ciclo viene stampato il contenuto del vettore **v** (però solo le ultime 4 posizioni).

Esercizio analisi (7)

Il seguente programma C compila correttamente? In caso affermativo, quali sono i valori stampati a tempo di esecuzione? (si motivi opportunamente la risposta data)

```c
#include <stdio.h>
#include <stdlib.h>
#define DIM 9

int Funz(char y[], int *N, int *M)
{
     int i;
     (*N)++;
     for (i=(*N)-1; i<*N; i++)
         y[i] = y[*M];
     return *N;
}
```

```c
int main () {
     char s[] = "Paperone";
     char ss[DIM]; int i=3, N = 2;
     N = (++N)-2;
     N = Funz(s, &N, &i);
     printf("N vale adesso: %d\n",N);
     {
         for (i=0; i<DIM; ++i) {
                  *(ss+i) = s[i];
                  printf("%c", *(ss+i));
         }
     }
     return 0;
}
```

Esercizio analisi (7)
Soluzione

Il programma compila correttamente e stampa:

```
N vale adesso: 2
Peperone
```

- La funzione **`Funz`** incrementa il valore riferito da **`N`**, e poi assegna all'elemento in posizione **`*N-1`** del vettore **`y`** il valore in posizione **`*M`**; restituisce infine il nuovo valore riferito da **`N`**
- Il **`main`** inizia dichiarando alcune vairabili, instanziando **`N`** a **`2`** e poi modificandolo con un assegnamento al valore **`1`**
- Viene poi invocata la funzione **`Funz`**, con parametri (**`"Paperone",1,3`**); per quanto detto, viene modificato l'elemento in posizione **`1`** (**`'a'`**), a cui viene assegnato il valore in posizione 3 (**`'e'`**)
- Inifine l'array **`s`** viene copiato (elemento per elemento) nell'array **`ss`**, che viene poi stampato a video

Esercizio Record di Attivazione (8)

Si consideri la seguente funzione:

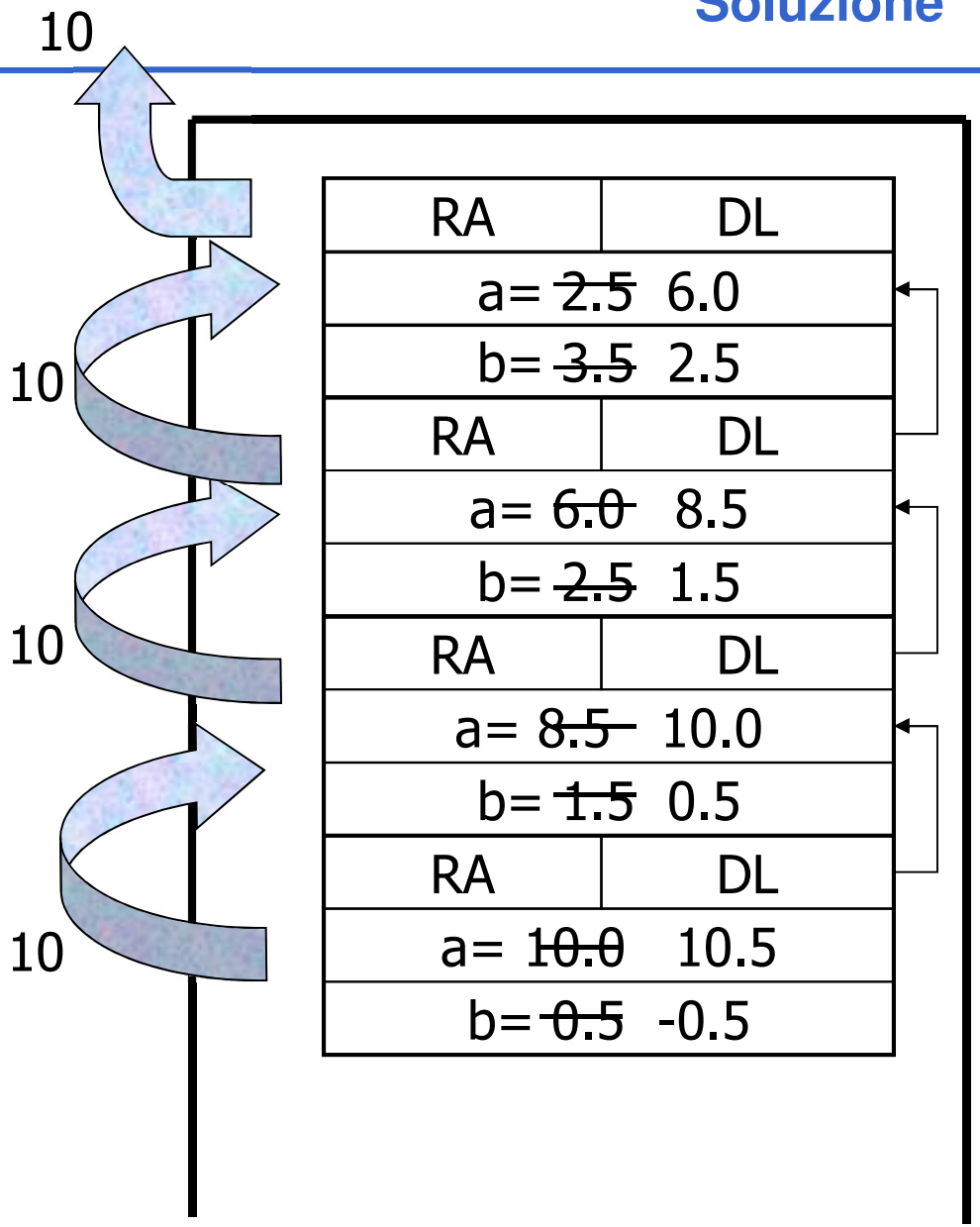
```
int funzione(float a, float b) {
    a = a+b;
    b--;
    if (b > 0)
       return funzione(a, b);
    else
       return a;
}
```

Si scriva il risultato della funzione quando invocata come:

```
funzione(2.5, 3.5)
```

e si disegnino i corrispondenti record di attivazione (si faccia particolare attenzione al fatto che la funzione restituisce un valore intero).

Esercizio Record di Attivazione (8)
Soluzione



Esercizio Record di Attivazione (9)

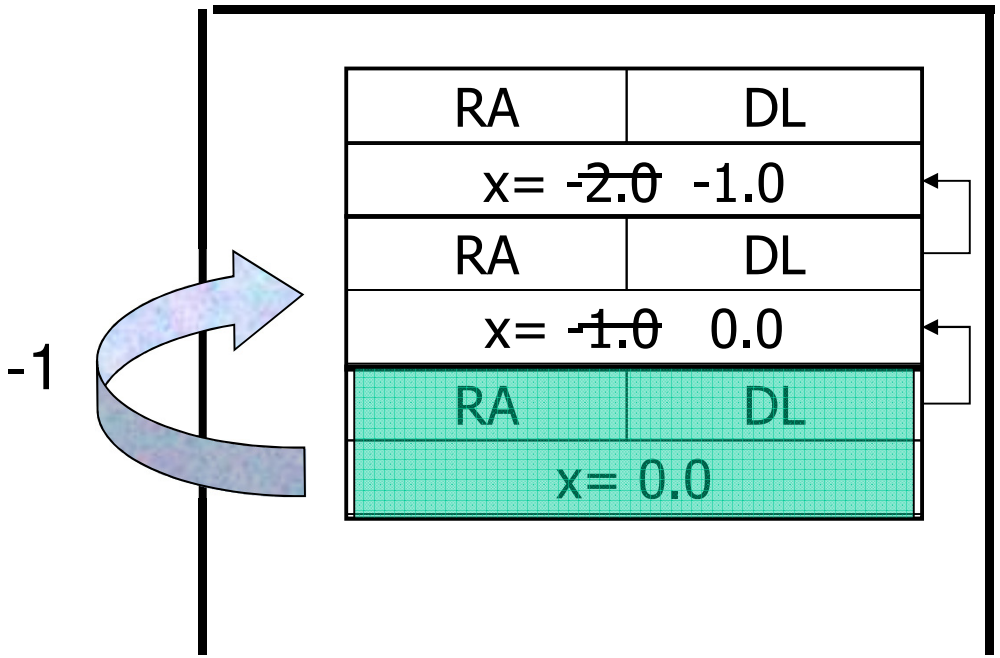
Si consideri la seguente funzione **W**:

```
double W(int x){
   if (x<0) {
          x++;
          return W(x)+W(x/2);
          x++;
   }
   else
          return -1;
}
```

Si scriva il risultato della funzione quando invocata come **W(-2)** e si disegnino i corrispondenti record di attivazione.

Esercizio Record di Attivazione (9)
Soluzione

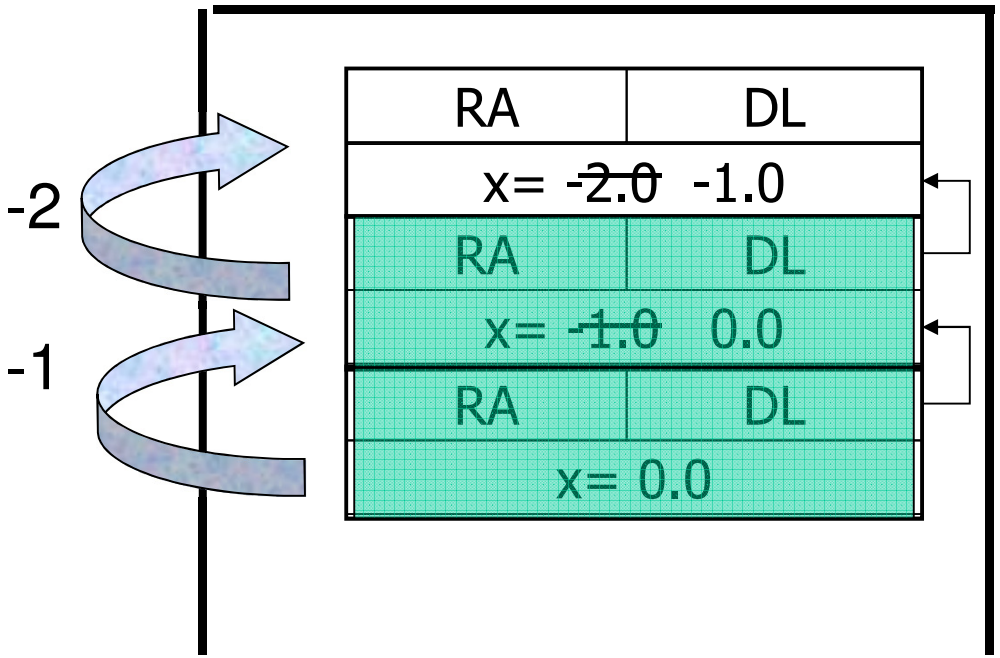
La funzione restituisce il valore –3.00. Supponendo che la valutazione degli addendi nella somma venga fatta a partire da sinistra, si ottiene prima:



poi l'eliminazione dell'ultimo record, e …

Esercizio Record di Attivazione (9)
Soluzione

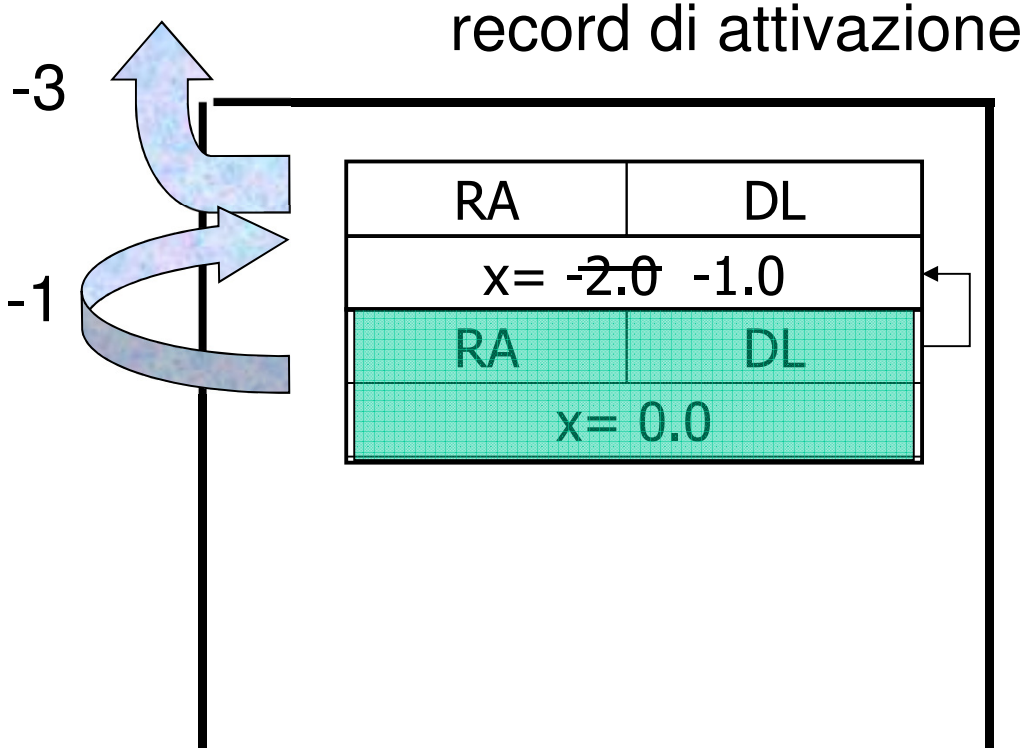
… dove viene fatta l'ultima invocazione di W(0), con somma finale restituita alla prima invocazione di W.



Quindi viene fatta la prima somma con risultato -2.00, restituita indietro

Esercizio Record di Attivazione (9)
Soluzione

…poi l'eliminazione dell'ultimo record, e un nuovo ulteriore record di attivazione per W(0):



La somma finale è restituita alla prima invocazione di W.

Esercizio sintesi (10)

Uno dei più antichi sistemi di codificazione di messaggi segreti si basa sulla sostituzione, secondo un certo ordine, dei caratteri componenti il messaggio.

Ad esempio, dato un messaggio composto dalle lettere:

{a, b, c}

E data una chiave di sostituzione che, per ogni lettera ne associa un'altra:

'a' → 'x'

'b' → 'y'

'c' → 'z'

Il messaggio originale può essere così riscritto:

{x, y, z}

Esercizio sintesi (10)

Si vuole costruire un sistema di codifica/decodifica di questo tipo, facendo le seguenti assunzioni:

1. Le lettere componenti il messaggio sono tutte minuscole, ed i messaggi non possono contenere altri caratteri che lettere (no spazi, no numeri)
2. Il codice di sostituzione è dato da un array di 26 caratteri, che viene interpretato nel seguente modo: nella posizione ad indice 0 vi è il carattere che deve sostituire la lettera ‘a’, in posizione con indice 1 vi è il carattere che deve sostituire la lettera ‘b’, etc.

Esercizio sintesi (10)

Si strutturi la soluzione implementando due funzioni:

```
void crypt( char source[],
            int length,
            char code[DIM_ALPHA],
            char dest[]);
```

```
void decrypt( char source[],
              int length,
              char code[DIM_ALPHA],
              char dest[]);
```

Ed infine si scriva un semplice main di prova.

Esercizio sintesi (10)
Soluzione

```
void crypt(char source[], int length, char
    code[DIM_ALPHA], char dest[]) {
       int i;

       for (i=0; i<length; i++) {
             dest[i] = code[source[i] - 'a'];
       }
}
```

Esercizio sintesi (10)
Soluzione

```
void decrypt(char source[], int length, char
    code[DIM_ALPHA], char dest[])
{
    int i;
    int j;
    int pos= -1;

    for (i=0; i<length; i++) {
       for (j=0; j<DIM_ALPHA && pos<0; j++) {
             if(source[i] == code[j])
                   pos = j;
       }
       dest[i] = 'a' + pos;
       pos = -1;
    }
}
```

Esercizio sintesi (10)
Soluzione

```
#define DIM 256
#define DIM_ALPHA 26

int main()
{
    char source[DIM] = "abc";
    char dest1[DIM] = {'\0',...};
    char dest2[DIM] = {'\0',...};
    char codice[DIM_ALPHA] = "cab";

    printf("ORIGINALE: %s\n", source);
    crypt(source, 3, codice, dest1);
    printf("CRIPTATO: %s\n", dest1);
    decrypt(dest1, 3, codice, dest2);
    printf("DE-CRIPTATO: %s\n", dest2);

    return 0; }
```

Esercizio sintesi (11)

Si vuole implementare un programma per il calcolo dell'inflazione su determinati prodotti commerciali.

A tal scopo ogni prodotto è rappresentato tramite una struttura **`item`**, definita da una stringa **`name`** con il nome del prodotto, e da due float **`old_price`** e **`new_price`** rappresentanti i prezzi.

Esercizio sintesi (11)

a) Si scriva una funzione **`lettura()`** che riceva come parametri di ingresso un vettore **`prezzi`** di strutture **`item`**, la dimensione fisica **`max`** del vettore **`prezzi`**, e un puntatore a intero **`num`** che rappresenta la dimensione logica del vettore. La funzione deve leggere da standard input il nome del prodotto ed i due prezzi, e deve copiare tale informazione nella prima posizione libera nel vettore **`prezzi`**.

Esercizio sintesi (11)

La funzione deve terminare se l'utente inserisce come nome del prodotto il termine "fine", oppure se viene raggiunta la dimensione fisica del vettore.

La dimensione logica del vettore **`prezzi`** così riempito deve essere restituita tramite il parametro **`num`** (passato appunto per riferimento). Al termine della lettura dei dati la funzione deve restituire il valore 0.

Esercizio sintesi (11)

b) Si scriva un programma **`main`** che, dopo aver definito un vettore di strutture **`item`** (di dimensione massima **`MAX_ITEM`**), invochi la funzione **`lettura()`** per riempire tale vettore.

Il programma stampi poi a video nome e tasso d'inflazione per ogni prodotto, utilizzando la formula:

$$infl_i = \left(\frac{new\_price_i}{old\_price_i} - 1 \right) * 100$$

Esercizio sintesi (11)
soluzione

```
#include <stdio.h>
#include <string.h>

#define DIM 21
#define MAX_ITEM 100

typedef struct {
     char name[DIM];
     float old_price;
     float new_price;
} item;
...
```

Esercizio sintesi (11)
soluzione

```
int lettura (item prezzi[], int max, int * num) {
     char name[DIM];
     *num = 0;

     printf("Inserire nome prodotto: ");   scanf("%s", name);

     while ((strcmp(name, "fine")) && (*num < max)) {
         strcpy(prezzi[*num].name, name);
         printf("Inserire old price: ");
         scanf("%f", &prezzi[*num].old_price);
         printf("Inserire new price: ");
         scanf("%f%*c", &prezzi[*num].new_price);

         (*num)++;

         printf("Inserire nome prodotto: ");
         scanf("%s", name);
     }   return 0;
}
```

Esercizio sintesi (11)
soluzione

```
int main() {
      item V[MAX_ITEM];
      int num, i, result;
      float infl;

      result = lettura(V, MAX_ITEM, &num);

      if (result!=0) {
          printf("Problemi durante la lettura...\n");
          exit(-1);
      }

      for (i=0; i < num; i++) {
          infl = (V[i].new_price/V[i].old_price -1)*100;
          printf("Inflazione del prodotto %s: %6.2f%%\n", V[i].name, infl);
      }
      return 0;
}
...
```