**Progetto d’esame**

# Specifica del Problema

Un alfabeto è un insieme finito di simboli. Un linguaggio finito su un alfabeto è un insieme finito di sequenze di lunghezza finita di simboli dell’alfabeto. Dato l’alfabeto {a, e, i, o, u}, scrivere un programma ANSI C che acquisisce da tastiera due linguaggi finiti su quel alfabeto e poi stampa sullo schermo l’unione dei due linguaggi e la differenza tra il primo linguaggio e il secondo. Almeno una delle due operazioni deve essere calcolata ricorsivamente.

# Analisi del Problema

## Dati di Input del problema

I dati di input del problema sono rappresentati da due insiemi di sequenze di caratteri appartenenti all’alfabeto {a,e,i,o,u}.

## Dati di Output del problema

I dati di output del problema sono rappresentati dall’ insieme ottenuto dalla differenza tra il primo e secondo linguaggio e da quello ottenuto dalla loro unione.

## Relazioni Intercorrenti tra i Dati

La differenza è un’operazione binaria, non commutativa, applicata a due insiemi, denotata dal simbolo “-“ , che come elemento neutro l’insieme vuoto *ε.* La differenza tra due insiemi contenuti in un insieme universo è definita come :

Dunque l’insieme degli elementi di A che non appartengono a B.

L’unione è un’operazione binaria, commutativa, associativa, applicata a due insiemi, denotata dal simbolo , che ha come elemento neutro l’insieme vuoto *ε.* L’unione tra due insiemi contenuti in un insieme universo è definita come:

Dunque l’insieme degli elementi che appartengono ad almeno uno dei due insiemi.

L’unione tra gli insiemi , può essere calcolata per via induttiva su nel seguente modo:

* se m= 0, allora
* se m > 0, allora per ogni *m* positivo, dove è il precedente di

Quindi la lunghezza di equivale a quella di B più quella della loro differenza.

# Progettazione dell’algoritmo

## Scelte di progetto

I linguaggi sono rappresentati come puntatori ad array di char . Gli array verranno allocati dinamicamente in quanto non si sa a priori quante siano le sequenze, o quanto siano lunghe, quindi per non porre limitazioni a rispetto della specifica del problema.

Per non porre limiti a rispetto della specifica del problema, un linguaggio può essere vuoto. Per creare un linguaggio vuoto bisogna inserire la lunghezza 0, quando verrà richiesta la lunghezza del linguaggio.

Il numero di sequenze per un linguaggio viene richiesto prima dell’inserimento delle sequenze stesse, per non limitare il programma, il numero di sequenze può essere pari a 0 ma non maggiore di 50, per evitare di utilizzare eccessiva memoria.

Qualora l’elaboratore utilizzato per eseguire il programma non dovesse possedere la memoria necessaria da allocare ai linguaggi, il programma segnalerà l’errore e terminerà senza eseguire i calcoli.

Una sequenza deve contenere un numero di caratteri minore di 100 e può essere vuota, nel programma quest’ultima è rappresentata dalla lettera ‘E’ maiuscola. In caso vengano inserite altre lettere nella stessa sequenza della ‘E’, essa non verrà accettata, inoltre se dovessero essere inserite più ‘E’ in una stessa sequenza, la sequenza viene ridotta ad una singola ‘E’ .

Essendo i linguaggi insiemi finiti di sequenze finite di lettere dell’alfabeto, uno stesso insieme non può contenere elementi uguali, dunque l’inserimento di una stessa sequenza all’interno di un linguaggio comporterà un errore, con la richiesta di una nuova acquisizione.

La stampa di un linguaggio vuoto non viene rappresentata come un insieme ma come la sola segnalazione che l’insieme ottenuto è vuoto.

I due insiemi calcolati dalla differenza e dall’unione dei due linguaggi sono rappresentati da due puntatori ad array di char, ad ogni sequenza viene allocata la memoria necessaria per contenere la sequenza di lunghezza massima del linguaggio.

## Passi dell’Algoritmo

I passi dell’algoritmo per la risoluzione del problema sono i seguenti:

* Acquisizione del numero di sequenze del primo linguaggio.
* Acquisizione delle sequenze del primo linguaggio.
* Acquisizione del numero di sequenze del secondo linguaggio.
* Acquisizione delle sequenze del secondo linguaggio.
* Calcolo della differenza tra il primo e secondo linguaggio.
* Calcolo ricorsivo dell’unione tra il primo e secondo linguaggio:

∗ Caso base: se il secondo insieme è vuoto, l’unione equivale al primo insieme.

∗ Caso induttivo: se il secondo insieme non è vuoto, viene spostato ricorsivamente l’ultimo elemento del secondo insieme a seguito dell’ultimo elemento del primo.

* Stampa dell’insieme ottenuto dalla differenza e dall’unione.

Per evitare ridondanza è stato necessario utilizzare dei sottoprogrammi per la stampa, allocazione di memoria e acquisizione delle sequenze.

# Implementazione dell’Algoritmo

Codice sorgente di unione\_e\_differenza\_linguaggi.c:



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* programma per differenza e unione tra due linguaggi \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* inclusione delle librerie \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* definizione delle costanti simboliche \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MAX\_SEQ 100 /\* lunghezza massima singola sequenza \*/

#define MAX\_LING 50 /\* numero massimo sequenze per linguaggio \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* dichiarazione delle funzioni \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int input\_n\_sequenze(int);

char \*\*alloca\_memoria(int,

int\*,

int);

int controllo\_sequenza(char \*\*,

int);

char \*\*input\_sequenze(int \*,

int);

char \*\*acquisici\_linguaggio(int \*,

int \*,

int \*,

int);

void differenza\_linguaggi(char \*\*,

char \*\*,

char \*\*,

int \*);

char \*unione\_linguaggi(char \*\*,

char \*\*,

int,

int);

void stampa\_linguaggi(char \*\*,

char \*\*,

char \*\*,

char \*\*,

int \*);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* definizione delle funzioni \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* definizione della funzione main \*/

int main(void)

{

/\* dichiarazione delle variabili locali alla funzione \*/

char \*\*linguaggio1, /\* input: sequenze del primo linguaggio \*/

\*\*linguaggio2, /\* input: sequenze del secondo linguaggio \*/

\*\*differenza, /\* output: sequenze ottenute dalla differenza \*/

\*\*unione; /\* output: sequenze ottenute dall'unione \*/

int n\_sequenze[3] = {0, /\* input: numero di sequenze per linguaggio \*/

0,

0},

i, /\* lavoro: variabile per cilco for \*/

errore\_mem = 0, /\* lavoro: controllo errori di allocazione memoria\*/

lunghezza\_max[3]; /\* lavoro: lunghezze delle sequenze più lunghe \*/

/\* stampa istruzioni per utente\*/

printf("\nProgramma per unione e differenza tra linguaggi.\n");

printf("\nLe sequenze possono solo contenere caratteri dell'alfabeto {a,e,i,o,u},");

printf("\nad eccezione della lettera 'E' che rappresenta la sequenza vuota.");

printf("\nAll'interno di un linguaggio non va ripetuta una stessa sequenza.\n");

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Acquisizione Linguaggi \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

linguaggio1 = acquisici\_linguaggio(lunghezza\_max,

n\_sequenze,

&errore\_mem,

0);

linguaggio2 = acquisici\_linguaggio(lunghezza\_max,

n\_sequenze,

&errore\_mem,

1);

/\* trova il massimo tra le due lunghezze \*/

if (lunghezza\_max[0] >= lunghezza\_max[1])

lunghezza\_max[2] = lunghezza\_max[0];

else

lunghezza\_max[2] = lunghezza\_max[1];

if (errore\_mem != 1)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Differenza \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* alloca memoria \*/

differenza = alloca\_memoria(n\_sequenze[0] + n\_sequenze[1],

&errore\_mem,

lunghezza\_max[2]);

if (errore\_mem != 1 &&

(n\_sequenze[0] != 0 ||

n\_sequenze[1] != 0))

{

/\* calcolo differenza \*/

differenza\_linguaggi(differenza,

linguaggio1,

linguaggio2,

n\_sequenze);

}

/\*\*\*\*\*\*\*\*\*\*/

/\* Unione \*/

/\*\*\*\*\*\*\*\*\*\*/

/\* alloca memoria \*/

unione = alloca\_memoria(n\_sequenze[0] + n\_sequenze[1],

&errore\_mem,

lunghezza\_max[2]);

if (errore\_mem != 1 &&

(n\_sequenze[0] != 0 ||

n\_sequenze[1] != 0))

{

/\* copia valori \*/

for (i = 0;

i < n\_sequenze[2];

i++)

strcpy(unione[i],

differenza[i]);

/\* se il secondo linguaggio e' vuoto non svolge i calcoli \*/

if (n\_sequenze[1] == 0)

{

for (i = 0;

i < n\_sequenze[0];

i++)

strcpy(unione[i],

linguaggio1[i]);

}

/\* se il primo linguaggio e' vuoto non svolge i calcoli \*/

if (n\_sequenze[0] == 0)

{

for (i = 0;

i < n\_sequenze[1];

i++)

strcpy(unione[i],

linguaggio2[i]);

}

/\* se i linguaggi non sono vuoti svolge i calcoli \*/

if (n\_sequenze[0] != 0 &&

n\_sequenze[1] != 0)

{

/\* calcolo unione \*/

unione\_linguaggi(unione,

linguaggio2,

n\_sequenze[2],

n\_sequenze[1]);

}

}

/\*\*\*\*\*\*\*\*/

/\*Stampa\*/

/\*\*\*\*\*\*\*\*/

stampa\_linguaggi(linguaggio1,

linguaggio2,

differenza,

unione,

n\_sequenze);

}

return 0;

}

/\* definizione della funzione per l'acquisizione del numero di sequenze da inserire \*/

int input\_n\_sequenze(int id\_linguaggio) /\* lavoro: identificatore per stampa \*/

{

/\* dichiarazione delle variabili locali alla funzione \*/

int esito\_lettura, /\* lavoro: variabile di controllo per scanf \*/

n\_sequenze; /\* output: numero di sequenze \*/

/\* acquisizione numero sequenze con validazione stretta \*/

printf("\n%s linguaggio: \n",

(id\_linguaggio == 0)?"Primo":"Secondo");

do

{

printf("Digitare il numero di sequenze da inserire (>= 0 e <100): ");

esito\_lettura = scanf("%d",

&n\_sequenze);

if (esito\_lettura != 1 ||

n\_sequenze < 0 ||

n\_sequenze > MAX\_LING)

printf("Valore non accettabile.\n");

while ( getchar() != '\n');

}

while (esito\_lettura != 1 ||

n\_sequenze < 0 ||

n\_sequenze > MAX\_LING);

return n\_sequenze;

}

/\* definizione della funzione per l'allocazione dinamica di memoria \*/

char \*\*alloca\_memoria(int n\_sequenze, /\* lavoro: numero di sequenze a cui allocare memoria \*/

int \*errore\_mem, /\* output: variabile di controllo errori allocazione\*/

int max\_seq) /\* lavoro: lunghezza massima sequenza \*/

{

/\* dichiarazione delle variabili locali alla funzione \*/

int i; /\* lavoro: variabile per ciclo for\*/

char \*\*linguaggio; /\* output: linguaggio a cui allocare memoria \*/

/\* allocazione puntatore a puntatore a char \*/

if (n\_sequenze == 0)

linguaggio = NULL;

else

{

linguaggio = calloc(n\_sequenze,

sizeof(char\*));

/\* allocazione scelta con controllo \*/

if (linguaggio != NULL)

{

for (i = 0;

i < n\_sequenze;

i++)

{

linguaggio[i] = calloc(max\_seq,

sizeof(char));

if (linguaggio[i] != NULL)

\*errore\_mem = 0;

else

\*errore\_mem = 1;

}

}

else

\*errore\_mem = 1;

}

/\* stampa errore se l'allocazione e' fallita \*/

if (\*errore\_mem == 1)

printf("\nImpossibile allocare memoria necessaria.");

return linguaggio;

}

/\* definizione funzione per controllo sequenze inserite \*/

int controllo\_sequenza(char \*\*linguaggio, /\* lavoro: linguaggio da controllare \*/

int id\_sequenza) /\* lavoro: numero sequenza da controllare \*/

{

/\* dichiarazione delle variabili locali alla funzione \*/

int i, /\* lavoro: variabile di controllo scanf \*/

lung\_sequenza = strlen(linguaggio[id\_sequenza]), /\* lavoro: lunghezza sequenza\*/

vuoto = 0, /\* lavoro: presenza carattere sequenza vuota \*/

esito\_ctrl = 1; /\* output: esito del controllo \*/

const char alfabeto[6] = {'a', /\* lavoro: alfabeto da utilizzare \*/

'e',

'i',

'o',

'u',

'E'};

/\* controllo rispetto lunghezza massima \*/

if (lung\_sequenza > (MAX\_SEQ - 1))

{

printf("\nValore non accettabile:");

printf("\nla sequenza supera il numero di caratteri consentito.");

esito\_ctrl = 0;

}

else

{

/\* controlli \*/

for ( i = 0;

i < lung\_sequenza;

i++ )

{

/\* controllo presenza lettere alfabeto o carattere sequenza vuota\*/

if ( linguaggio[id\_sequenza][i] != alfabeto[0] &&

linguaggio[id\_sequenza][i] != alfabeto[1] &&

linguaggio[id\_sequenza][i] != alfabeto[2] &&

linguaggio[id\_sequenza][i] != alfabeto[3] &&

linguaggio[id\_sequenza][i] != alfabeto[4] &&

linguaggio[id\_sequenza][i] != alfabeto[5])

{

printf("\nValore non accettabile:");

printf("\nla sequenza non contiene solo caratteri dell'alfabeto.");

esito\_ctrl = 0;

}

else

{

/\* controllo numero di 'E' nella sequenza \*/

if (linguaggio[id\_sequenza][i] == alfabeto[5])

vuoto++;

}

}

/\* controllo se 'E' sia l'unico carattere della sequenza \*/

if (esito\_ctrl != 0 &&

vuoto != 0 &&

vuoto != lung\_sequenza)

{

printf("\nValore non accettabile:");

printf("\nla sequenza vuota puo' contenere unicamente la lettera 'E'");

esito\_ctrl = 0;

}

}

/\* se una sequenza contiene piu' 'E' viene ridotta ad 'E' \*/

if (esito\_ctrl != 0 &&

linguaggio[id\_sequenza][0] == 'E')

strcpy(linguaggio[id\_sequenza],"E");

/\* controllo unicita' sequenza \*/

if ( esito\_ctrl == 1)

{

for (i = 0;

i < id\_sequenza;

i++ )

{

if ( strcmp(linguaggio[id\_sequenza],

linguaggio[i]) == 0 )

{

printf("\nValore non accettabile:");

printf("\nla sequenza e' gia' stata inserita.");

esito\_ctrl = 0;

}

}

}

return esito\_ctrl;

}

/\* definizione funzione per l'acquisizione delle sequenze \*/

char \*\*input\_sequenze(int \*lunghezza\_max, /\* lavoro: lunghezza della sequenze piu' lunga \*/

int n\_sequenze) /\* lavoro: numero di sequenze da richiedere \*/

{

/\* dichiarazione delle variabili locali alla funzione \*/

int i, /\* lavoro: variabile per ciclo for \*/

esito\_lettura, /\* lavoro: variaible controllo scanf \*/

esito\_ctrl; /\* lavoro: variabile esito controllo su sequenza \*/

char \*\*linguaggio = calloc(n\_sequenze, /\* output: insieme di sequenze da inserire\*/

sizeof(char \*));;

if (linguaggio != NULL)

{

/\* acquisici le sequenze \*/

for (i = 0;

i < n\_sequenze;

i++)

{

/\* validazione stretta \*/

do

{

printf("\nInserire sequenza numero %d (con lunghezza <100 e >0): ",

i+1);

esito\_lettura = scanf("%ms",

&linguaggio[i]);

esito\_ctrl = controllo\_sequenza(linguaggio,

i);

if ( esito\_lettura != 1 )

printf("\nL'acquisizione non e' avvenuta con successo.");

while (getchar() != '\n');

}

while (esito\_lettura != 1 ||

esito\_ctrl != 1 );

/\* contollo sequenza di lunghezza maggiore \*/

if (\*lunghezza\_max < strlen(linguaggio[i]))

\*lunghezza\_max = strlen(linguaggio[i]);

}

}

return linguaggio;

}

/\* definizione funzione per acquisire un linguaggio \*/

char \*\*acquisici\_linguaggio(int \*lunghezza\_max,

int \*n\_sequenze,

int \*errore\_mem,

int id\_linguaggio)

{

/\* dichiarazione delle variabili locali alla funzione \*/

char \*\*linguaggio = NULL; /\* output: linguaggio da acquisire \*/

/\* acquisizione numero sequenze \*/

n\_sequenze[id\_linguaggio] = input\_n\_sequenze(id\_linguaggio);

if (\*errore\_mem != 1)

{

linguaggio = input\_sequenze(lunghezza\_max,

n\_sequenze[id\_linguaggio]);

}

return linguaggio;

}

/\* definizione funzione per calcolare la differenza tra il primo e il secondo linguaggio \*/

void differenza\_linguaggi(char \*\*differenza, /\* output: insieme da calcolare \*/

char \*\*linguaggio1, /\* lavoro: primo linguaggio \*/

char \*\*linguaggio2, /\* lavoro: secondo linguaggio \*/

int \*n\_sequenze) /\* lavoro: numero sequenze per linguaggio \*/

{

/\* dichiarazione delle variabili locali alla funzione \*/

int i, /\* lavoro: variabile per ciclo for \*/

j, /\* lavoro: variabile per ciclo for \*/

disuguaglianza = 0; /\* lavoro: contatore disuguaglianza tra linguaggi \*/

/\* calcolo differenza \*/

for (i = 0;

i < n\_sequenze[0];

i++)

{

for (j = 0;

j < n\_sequenze[1];

j++)

{

if (strcmp(linguaggio1[i],

linguaggio2[j]) != 0 )

disuguaglianza ++;

}

if (disuguaglianza == n\_sequenze[1])

{

differenza[n\_sequenze[2]] = linguaggio1[i];

n\_sequenze[2]++;

}

disuguaglianza = 0;

}

}

/\* dichiarazione funzione per unione dei due linguaggi \*/

char \*unione\_linguaggi(char \*\*linguaggio1, /\* lavoro: primo linguaggio da unire \*/

char \*\*linguaggio2, /\* lavoro: secondo linguaggio da unire \*/

int n, /\* lavoro: numero sequenze primo linguaggio \*/

int m) /\* lavoro: numero sequenze secondo linguaggio \*/

{

/\* caso induttivo \*/

if ( m != 0)

{

strcpy(linguaggio1[n],

unione\_linguaggi(linguaggio1,

linguaggio2,

n + 1,

m - 1));

}

return linguaggio2[m];

}

/\* dichiarazione funzione per stampa linguaggi \*/

void stampa\_linguaggi(char \*\*linguaggio1, /\* lavoro: primo linguaggio \*/

char \*\*linguaggio2, /\* lavoro: secondo linguaggio \*/

char \*\*differenza, /\* lavoro: linguaggio da stampare \*/

char \*\*unione, /\* lavoro: linguaggio da stampare \*/

int \*n\_sequenze) /\* lavoro: numero di sequenze dei linguaggi \*/

{

/\* dichiarazione delle variabili locali alla funzione \*/

int i, /\* lavoro: variabile per ciclo for \*/

j, /\* lavoro: variabile per ciclo for \*/

sequenze[2], /\* lavoro: numero di sequenze da stampare \*/

id\_stampa; /\* lavoro: id linguaggio da stampare \*/

/\* se entrambi i linguaggi sono vuoti \*/

if (n\_sequenze[0] == 0 &&

n\_sequenze[1] == 0)

printf("\nGli insiemi ottenuti dalla differenza e unione sono vuoti\n");

else

{

/\* se il primo linguaggio e' vuoto \*/

if (n\_sequenze[0] == 0)

{

printf("\nL'insieme ottenuto dalla differenza e' vuoto");

sequenze[0] = n\_sequenze[1] + n\_sequenze[2];

id\_stampa = 1;

}

/\* se il secondo linguaggio e' vuoto \*/

if ( n\_sequenze[1] == 0)

{

sequenze[0] = n\_sequenze[0];

sequenze[1] = n\_sequenze[0];

id\_stampa = 2;

}

/\* se nessun linguaggio e' vuoto \*/

if (n\_sequenze[0] != 0 &&

n\_sequenze[1] != 0 )

{

sequenze[0] = n\_sequenze[2] + n\_sequenze[1];

sequenze[1] = n\_sequenze[2];

id\_stampa = 2;

}

for (i = 0;

i < id\_stampa;

i++)

{

/\* stampa insiemi \*/

printf("\nInsieme ottenuto %s = {",

(i == 1)?"dalla differenza":"dall'unione");

for (j = 0;

j < sequenze[i];

j++)

{

printf("%s",

(i == 1)?differenza[j]:unione[j]);

if (j != (sequenze[i] - 1))

putchar(',');

}

printf("}");

}

printf("\n");

}

}

Makefile:

unione\_e\_differenza\_linguaggi: unione\_e\_differenza\_linguaggi.c makefile

gcc -ansi -Wall -O unione\_e\_differenza\_linguaggi.c -o unione\_e\_differenza\_linguaggi

pulisci:

rm -f unione\_e\_differenza\_linguaggi.o

pulisci\_tutto:

rm -f unione\_e\_differenza\_linguaggi unione\_e\_differenza\_linguaggi.o

# Testing del Programma

I seguenti test mostrano che il programma accetta un numero di sequenze necessariamente compreso tra 0 e 50, e che le sequenze possono contenere solo caratteri dell’alfabeto in numero minore di 100, ad eccezione del carattere ‘E’ che rappresenta la sequenza vuota.

## Test 1

Numero sequenze primo linguaggio: 2

Sequenze primo linguaggio: { a,e} Numero sequenze linguaggio: 3 Sequenze secondo linguaggio: { i,o,u} Differenza: { a,e} Unione: {a,e,u,o,i}

## Test 2

Numero sequenze primo linguaggio: 0

Sequenze primo linguaggio: { } Numero sequenze linguaggio: 3 Sequenze secondo linguaggio: { i,o,u} Differenza: { } Unione: {u,o,i}

## Test 3

Numero sequenze primo linguaggio: 3

Sequenze primo linguaggio: { a,e,i } Numero sequenze secondo linguaggio: 0 Sequenze secondo linguaggio: { } Differenza: { a,e,i } Unione: { a,e,i }

## Test 4

Numero sequenze primo linguaggio: 1

Sequenze primo linguaggio: { E } Numero sequenze secondo linguaggio: 0 Sequenze secondo linguaggio: { E } Differenza: { } Unione: { E }

## Test 5

Numero sequenze primo linguaggio: 5

Sequenze primo linguaggio: { aoao,ioa,ieio,o,iiii } Numero sequenze secondo linguaggio: 2 Sequenze secondo linguaggio: { uoii,iiii } Differenza: { aoao,ioa,ieio,o} Unione: { aoao,ioa,ieio,o,iiii,uoii }

## Test 6

Numero sequenze primo linguaggio: 7

Sequenze primo linguaggio: { aooiii,iaoao,uiuoai,iaioiaaioi,uuauaoaioi,ioao,E } Numero sequenze secondo linguaggio: 6 Sequenze secondo linguaggio: { aooiii,iaoao,uiuoai,iaioiaaioi,uuauaoaioi,ioao } Differenza: { E } Unione: {aooiii,iaoao,uiuoai,iaioiaaioi,uuauaoaioi,ioao,E }

## Test 7

Numero sequenze primo linguaggio: 3

Sequenze primo linguaggio: { iiiiiiiiiiiiiiii,eeee,aaaaaaaa } Numero sequenze secondo linguaggio: 2 Sequenze secondo linguaggio: { oaoaoaoaaoa,ieieieieie } Differenza: { iiiiiiiiiiiiiiii,eeee,aaaaaaaa } Unione: { iiiiiiiiiiiiiiii,eeee,aaaaaaaa, }

## Test 8

Numero sequenze primo linguaggio: 26

Sequenze primo linguaggio: { E, aa,ae,ai,ao,au,ea,ee,ei,eo,eu,ia,ie,ii,io,iu,oa,oe,oi,oo,ou,ua,ue,ui,uo,uu } Numero sequenze secondo linguaggio: 5 Sequenze secondo linguaggio: { aa,ee,ii,oo,uu } Differenza: { E,ae,ai,ao,au,ea,ei,eo,eu,ia,ie,io,iu,oa,oe,oi,ou,ua,ue,ui,uo} Unione: { E, aa,ae,ai,ao,au,ea,ee,ei,eo,eu,ia,ie,ii,io,iu,oa,oe,oi,oo,ou,ua,ue,ui,uo,uu }

## Test 9

Numero sequenze primo linguaggio: 4

Sequenze primo linguaggio: { oooeooeoaiaiaiaiaaiaoaoaoao,oo,ieoooa,oeeieoaoeieoeaoeoa } Numero sequenze secondo linguaggio: 5 Sequenze secondo linguaggio: { oooeooeoaiaiaiaiaaiaoaoaoao,iuiuuuuu,aeaeiaiai,oo,ieoooa } Differenza: { oeeieoaoeieoeaoeoa } Unione: { oooeooeoaiaiaiaiaaiaoaoaoao,oo,ieoooa,oeeieoaoeieoeaoeoa, aeaeiaiai,iuiuuuuu }

## Test 10

Numero sequenze primo linguaggio: 10

Sequenze primo linguaggio: {uiiioio,ioeoeoeiao,ieieoa,oeoeoeooo, eoeo,i,oeoeoeoaoaaoa,eeea,ooiaaa,o}   Sequenze secondo linguaggio: { ioeoeoeiao,ieieoa,ooiaaa ,i,aaaiaaa,oioioi} Differenza: { uiiioio ,oeoeoeooo,eoeo,oeoeoeoaoaaoa ,o } Unione: { uiiioio,ioeoeoeiao,ieieoa,oeoeoeooo, eoeo,i,oeoeoeoaoaaoa,eeea,ooiaaa,o, oioioi,aaaiaaa }

# Verifica del Programma

## Estratto di codice scelto

    /\* trova il massimo tra le due lunghezze \*/

    if (lunghezza\_max[0] >= lunghezza\_max[1])

        lunghezza\_max[2] = lunghezza\_max[0];

    else

        lunghezza\_max[2] = lunghezza\_max[1];

## Proprietà da verificare

Se denotiamo con:

* *x* la variabile *lunghezza\_max[0]*;
* *y* la variabile *lunghezza\_max[1]*;
* *z* la variabile *lunghezza\_max[2]*;

Per verificarne la correttezza la proprietà da verificare sarà quella dettata dalle Regole di Dijkstra nelle istruzioni di selezione, indicando con l’espressione “ ” con l’istruzione “;” e con l’istruzione “;”.La proprietà da verificare è:

## Sviluppo

Il brano di codice è sempre corretto perché, formalizzando la postcondizione *R* nel seguente modo: