**RELAZIONE Traccia - Lettura numerica**

Si vuole implementare un programma per la simulazione del gioco della lettura numerica.   
Si sceglie qualsiasi numero nella tabella in figura. Scandendo il nome in inglese lettera dopo lettera si prendono i numeri corrispondenti alle lettere sottraendo quelli nelle caselle gialle e sommando quelli nelle caselle bianche.   
Il risultato sarà il numero scelto con il segno negativo o positivo.   
Ad esempio, TWENTY-TWO risulta: 20 – 25 – 4 – 2 + 20 + 11 + 20 – 25 + 7 = 22

|  |  |  |  |
| --- | --- | --- | --- |
| E         4 | I       17 | N         2 | S       16 |
| L       24 | F         9 | T       20 | R         6 |
| W       25 | U       12 | G       22 | O         7 |
| V         1 | X       27 | Y       11 | M         3 |

Sviluppare un programma che dato un numero in lettere permette di attraversare la tabella visualizzando le posizioni lettera per lettera e il risultato finale.

**Codice sorgente – main.c**

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#include "lib.h"

/\*

Lettura numerica

Il programma implementerà una simulazione del gioco della lettura numerica.

Scelto un qualsiasi numero nella Tabella (disponibile nella cartella "Contenuti\_Multimediali"),

il programma scandirà il nome in inglese lettera dopo lettera prendendo i numeri corrispondenti alle lettere,

sottraendo quelli nelle caselle gialle e sommando quelli nelle caselle bianche.

Il risultato sarà il numero scelto, con il segno negativo o positivo.

Il programma, dato un numero in lettere e in inglese, permetterà di attraversare la tabella visualizzando/stampando le posizioni lettera per lettera e il risultato finale.

Ad esempio, TWENTYTWO risulta: 20-25-4-2+20+11+20-25+7= 22.

\*/

// Dichiarazione Tabella

lettera\_singola tab[4][4]={

{{'E', 4, 1}, {'I', 17, 0}, {'N', 2, 1}, {'S', 16, 0}},

{{'L', 24, 0}, {'F', 9, 1}, {'T', 20, 0}, {'R', 6, 1}},

{{'W', 25, 1}, {'U', 12, 0}, {'G', 22, 1}, {'O', 7, 0}},

{{'V', 1, 0}, {'X', 27, 1}, {'Y', 11, 0}, {'M', 3, 1}}

};

**int** main(){

**char** nome\_num[20];

stampa\_tab(tab);

printf("Immettere il nome del numero (presente in tabella e in inglese) che vuoi leggere: ");

scanf("%s", nome\_num);

strupr(nome\_num);

**while**(check\_num\_tab\_ing(nome\_num) != 1){

printf("Immettere correttamente il nome del numero, assicurandosi che sia in tabella e sia scritto in inglese:");

scanf("%s", nome\_num);

strupr(nome\_num);

}

scandisci\_lettere(nome\_num, tab);

**return 0**;

}

**Codice sorgente – lib.h**

// Dichiarazione struct

**typedef struct**{

**char** carattere;

**int** valore;

**int** colore; // 1 se giallo, 0 se bianco

}lettera\_singola;

// Dichiarazione delle funzioni

**int** check\_num\_tab\_ing(**char**[]);

**void** scandisci\_lettere(**char**[], lettera\_singola[][4]);

**void** stampa\_tab(lettera\_singola[][4]);

**void** trova\_tab(**char**, lettera\_singola[][4]);

**int** segno\_lett(lettera\_singola);

**Codice sorgente – lib.c**

#include "lib.h"

// Funzione che verificherà se la stringa del numero in lettere è presente nella tabella e in inglese.

// Parametri in INPUT: char numero[], stringa del numero in lettere.

// Parametri in OUTPUT: ritorna 1 se la stringa rappresenta un numero presente nella tabella

// ritorna 0 altrimenti.

**int** check\_num\_tab\_ing(**char** numero[]){

if(strcmp(numero, "FOUR") == 0 || strcmp(numero, "SEVENTEEN") == 0 || strcmp(numero, "TWO") == 0 || strcmp(numero, "SIXTEEN") == 0 || strcmp(numero, "TWENTYFOUR") == 0 || strcmp(numero, "NINE") == 0 || strcmp(numero, "TWENTY") == 0 || strcmp(numero, "SIX") == 0 || strcmp(numero, "TWENTYFIVE") == 0 || strcmp(numero, "TWELVE") == 0 || strcmp(numero, "TWENTYTWO") == 0 || strcmp(numero, "SEVEN") == 0 || strcmp(numero, "ONE") == 0 || strcmp(numero, "TWENTYSEVEN") == 0 || strcmp(numero, "ELEVEN") == 0 || strcmp(numero, "THREE") == 0)

**return 1**;

**return** **0**;

}

// Prendo numero come parola, prendo le singole lettere scorrendo z, confronto z-lettera con tab[i][j].carattere, // scorrendo i e j,

// finchè non trovo che tab[i][j].carattere coincide con la singola lettera, poi tramite la funzione

// trova\_tab(numero[z], tab) cercherò e stamperò la posizione in tabella dove la lettera numero[z] coincide con

// tab[i][j].carattere, e tramite segno\_lett(tab[i][j]) ritornerò il valore della lettera in tabella (tab[i][j].valore) e il suo // segno dipendentemente da tab[i][j].colore, e infine svolgerò la somma, tramite il ciclo while farò ciò per ogni

// lettera, e infine stamperò il risultato.

// Parametri in INPUT: char numero[], stringa del numero in lettere.

// lettera\_singola tab[][4], la tabella/array bidimensionale di tipo struct lettera\_singola.

// Parametri in OUTPUT: nessuno.

**void** scandisci\_lettere(**char** numero[], lettera\_singola tab[][4]){

**int** z=0, i=0, j=0, somma= 0;

**while**(numero[z] != '\0'){

**for**(i=0; i<4; i++){

**for**(j=0; j<4; j++){

**if**(numero[z] == tab[i][j].carattere){

trova\_tab(numero[z], tab);

somma= somma + segno\_lett(tab[i][j]);

}

}

}

z++;

}

printf("Risultato finale: %d", somma);

}

// Semplice funzione che stamperà a schermo la tabella (in input) con i rispettivi caratteri, valori e colori, con

// \033[33mtesto\033[0m che mi formatta il testo di giallo.

// Parametri in INPUT: lettera\_singola tab[][4], la tabella/array bidimensionale di tipo struct lettera\_singola.

// Parametri in OUTPUT: nessuno.

**void** stampa\_tab(lettera\_singola tab[][4]){

**int** i, j;

printf("Tabella stampata:\n");

**for**(i=0; i<4; i++){

**for**(j=0; j<4; j++){

**if**(tab[i][j].colore == 1)

printf("[\033[33m%c %d\033[0m]", tab[i][j].carattere, tab[i][j].valore);

**else**

printf("[%c %d]", tab[i][j].carattere, tab[i][j].valore);

}

printf("\n");

}

printf("\n");

}

// Scorro e stampo la tabella (in input) indicando con "->" la posizione della singola lettera (in input).

// Parametri in INPUT: char lettera, un carattere (che sarà numero[z]).

// lettera\_singola tab[][4], la tabella/array bidimensionale di tipo struct lettera\_singola.

// Parametri in OUTPUT: nessuno.

**void** trova\_tab(**char** lettera, lettera\_singola tab[][4]){

**int** i, j, i\_t= 0, j\_t= 0;

printf("\nLettera trovata:\n");

**for**(i=0; i<4; i++){

**for**(j=0; j<4; j++){

**if**(tab[i][j].colore == 1){

**if**(lettera == tab[i][j].carattere){

printf("[->\033[33m%c %d\033[0m]", tab[i][j].carattere, tab[i][j].valore);

i\_t= i;

j\_t= j;

}

**else**

printf("[\033[33m%c %d\033[0m]", tab[i][j].carattere, tab[i][j].valore);

}

**else**{

**if**(lettera == tab[i][j].carattere){

printf("[->%c %d]", tab[i][j].carattere, tab[i][j].valore);

i\_t= i;

j\_t= j;

}

**else**

printf("[%c %d]", tab[i][j].carattere, tab[i][j].valore);

}

}

printf("\n");

}

printf("\t\t\t\tin posizione riga %d e colonna %d.", i\_t+1, j\_t+1);

printf("\n");

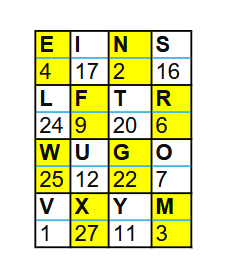
}

// Funzione che semplicemente associa il segno giusto al valore della lettera in tabella (elem\_tab aka tab[i][j]), in // base al suo colore.

// Parametri in INPUT: lettera\_singola elem\_tab, (che sarà lettera\_singola tab[i][j]).

// Parametri in OUTPUT: -(elem\_tab.valore) sse elem\_tab.colore == 1

// elem\_tab.valore sse elem\_tab.colore != 1

****

**int** segno\_lett(lettera\_singola elem\_tab){

**if**(elem\_tab.colore == 1)

**return -(elem\_tab.valore)**;

**else**

**return elem\_tab.valore**;

}

**TEST DI ESECUZIONE**

**Immagine che contiene testo, schermata, Carattere

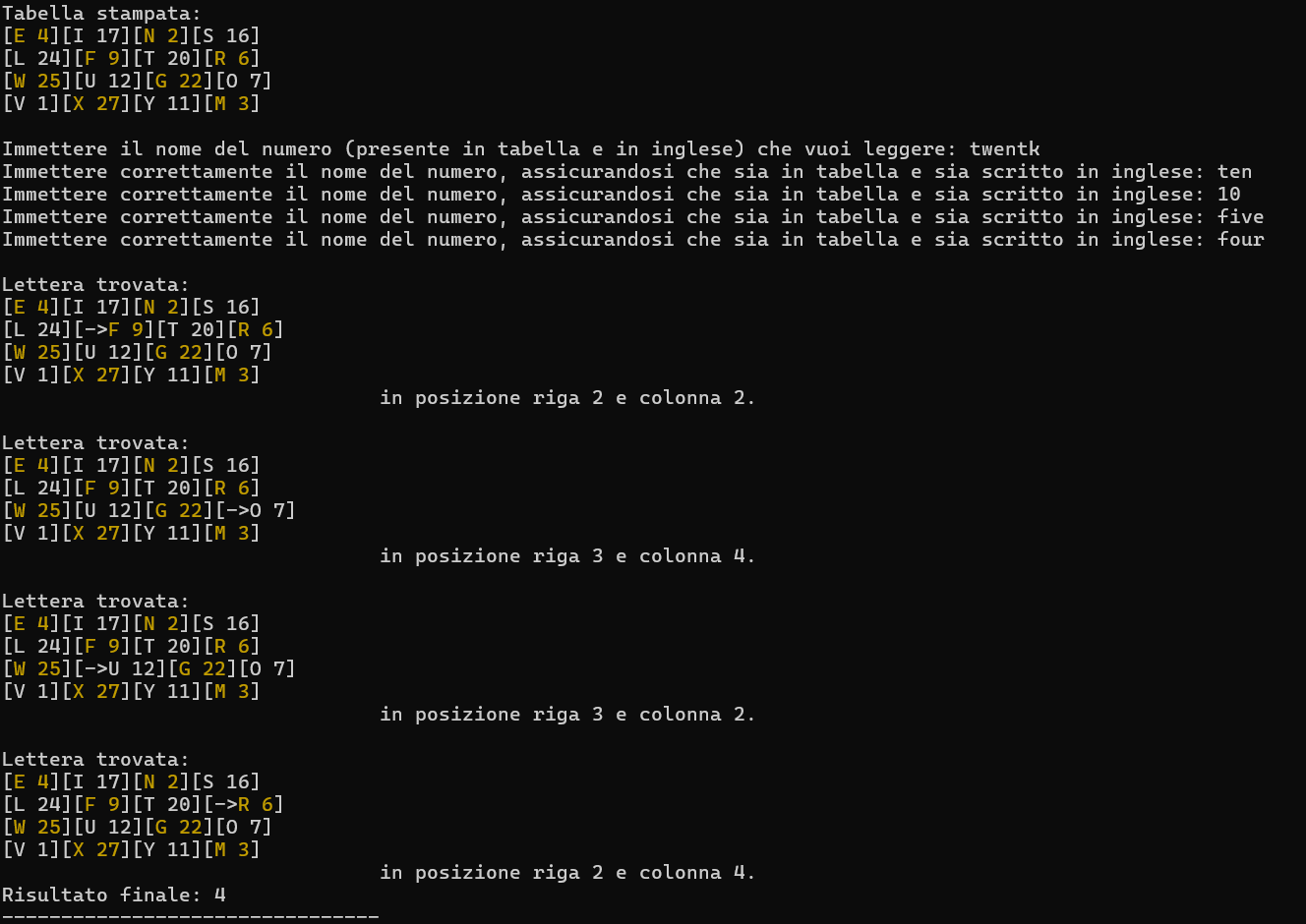
Descrizione generata automaticamente1° TEST**

Il programma chiede di inserire il nome di un numero, in inglese, e presente all'interno della tabella.

Inserito “twelve”, il programma mi mostrerà la posizione in tabella delle varie lettere (con “->”) e mi stamperà l’indice di riga e colonna per ogni lettera.

Alla fine, restituirà un risultato finale (12, che coincide col nome del numero), dopo che ha calcolato la somma dei vari valori di ogni lettera, con segno dipendente dal proprio colore.

**2° TEST**

****

Il programma chiede di inserire il nome di un numero, in inglese, e presente all'interno della tabella.

Ho simulato un errore di inserimento, prima ortografico, poi inserendo il nome di un numero non presente in tabella, poi inserendo un numero e infine di nuovo il nome di un numero assente.

Il programma mi ha chiesto di inserire correttamente il nome di un numero presente in tabella e in inglese, finché non ha ricevuto un input corretto (four).

Inserito “four”, il programma mi mostrerà la posizione in tabella delle varie lettere (con “->”) e mi stamperà l’indice di riga e colonna per ogni lettera.

Alla fine, restituirà un risultato finale (4, che coincide col nome del numero), dopo che ha calcolato la somma dei vari valori di ogni lettera, con segno dipendente dal proprio colore.

****

**3° TEST**

Il programma chiede di inserire il nome di un numero, in inglese, e presente all'interno della tabella.

Ho inserito il nome di un numero presente in tabella ma con alcune lettere maiuscole e altre minuscole.

Inserito “SeVeN”, il programma si mostrerà “case insensitive”, accettando l’input.

Mi mostrerà la posizione in tabella delle varie lettere (con “->”) e mi stamperà l’indice di riga e colonna per ogni lettera.

Alla fine, restituirà un risultato finale (7, che coincide col nome del numero), dopo che ha calcolato la somma dei vari valori di ogni lettera, con segno dipendente dal proprio colore.

**Traccia - Dizionario**

Si vuole simulare la gestione di un dizionario della lingua italiana.   
Il dizionario è indicizzato tramite le lettere dell’alfabeto ed ogni parola contiene i sinonimi (massimo 5) e la sua spiegazione.

Permettere all’utente di:

* Inserire una parola (mantenere l’ordine alfabetico)
* Ricercare una parola nel dizionario e visualizzare il significato

Provvedere all’implementazione dell’algoritmo per la simulazione del dizionario.

**Codice sorgente – main.c**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "lib.h"

/\*

DIZIONARIO

Il programma simulerà la gestione di un dizionario della lingua italiana.

Con un array indice di 26 elementi di tipo struct dizionario, ogni elemento avrà un campo lettera

(carattere che rappresenterà la lettera dell'indice) ed un campo parole[30] di tipo struct parola, tale che ogni elemento di indice con una lettera specifica avrà la possibilità di immagazzinare 30 parole diverse nell'array parole[30] di tipo struct parola.

Ogni elemento dell'array parole di quella lettera (quindi ogni parola indicizzata) potrà accedere al campo char parola\_sinonimi[6][20] (un'array bidimensionale dove inserirò la parola principale e 5 sinonimi, di lunghezza massima 20 caratteri) e un campo char significato[50] (stringa dove inserirò la spiegazione della parola di lunghezza massima 50 caratteri).

Sarà possibile tramite due funzioni: -inserire una parola indicizzandola. -ricercare una parola nel dizionario e visualizzare il significato.

E la scelta dell'operazione sarà proposta all'utente e in loop finché non inserirà "3".

\*/

**int** main() {

// Indice del nostro dizionario

dizionario indice[26];

iniz\_indice(indice);

stampa\_indice(indice);

operazione(scelta(), indice);

**return 0**;

}

**Codice sorgente – lib.h**

// Dichiarazione struct

**typedef struct**{

**char** parola\_sinonimi[6][20];

**char** significato[50];

}parola;

**typedef struct**{

**char** lettera;

parola parole[30];

}dizionario;

// Dichiarazione delle funzioni

**void** stampa\_indice(dizionario[]);

**void** iniz\_indice(dizionario[]);

**int** scelta();

**void** operazione(int, dizionario[]);

**void** ins\_parola(dizionario[]);

**void** cerca\_parola(dizionario[]);

**Codice sorgente – lib.c**

#include "lib.h"

#include <stdio.h>

// Stampa indice, stampa il campo .lettera di ogni elemento di dizionario indice[26], quindi indice[i].lettera.

// Parametri in INPUT: dizionario indice[] (array di 26 elementi di tipo struct dizionario).

// Parametri in OUTPUT: nessuno.

**void** stampa\_indice(dizionario indice[]){

**int** i;

printf("Indice del dizionario:\n");

**for**(i=0; i<26; i++){

printf("-%c ", indice[i].lettera);

}

}

// Inizializzazione dell'indice, assegna una lettera dell'alfabeto ad ogni elemento di dizionario indice[26], nel suo campo lettera.

// Parametri in INPUT: dizionario indice[] (array di 26 elementi di tipo struct dizionario).

// Parametri in OUTPUT: nessuno.

**void** iniz\_indice(dizionario indice[]){

**int** i, j;

**for**(i=0; i<26; i++)

indice[i].lettera= 'A'+i;

}

// Scelta dell'operazione da effettuare, se verrà immessa una scelta non disponibile, la funzione chiederà con un // ciclo while di riprovare finché non riceverà 1, 2 o 3.

// Parametri in INPUT: nessuno.

// Parametri in OUTPUT: int op (ovvero l'operazione scelta dall'utente e inserita tramite tastiera).

**int** scelta(){

**int** op;

printf("\n\nChe operazione vuoi effettuare:\n1) Inserire una parola\n2) Cercare una parola\n3) Spegni\n\n");

scanf("%d", &op);

**while**(op != 1 && op != 2 && op != 3){

printf("RIPROVARE: ");

scanf("%d", &op);

}

**return op**;

}

// Operazione, se op sarà 1, richiamerà la funzione ins\_parola(indice), se sarà op sarà 2 richiamerà la funzione

// cerca\_parola(indice), se op né 1 né 2 (if non chiede esplicitamente che sia 3, perché op= 3 sarà forzato dalla

// funzione scelta() nel caso non si immetta 1 o 2) il programma non richiamerà nessuna funzione e terminerà.

// Parametri in INPUT: dizionario indice[] (array di 26 elementi di tipo struct dizionario).

// int op (ovvero l'output di scelta(), e indicherà l'operazione scelta e immessa dall'utente

// da tastiera.

// Parametri in OUTPUT: nessuno.

**void** operazione(**int** op, dizionario indice[]){

**if**(op == 1)

ins\_parola(indice);

**else if**(op == 2)

cerca\_parola(indice);

**else**

**return**;

}

// Inserimento parola nel dizionario con eventuali sinonimi e significato.

// La funzione chiede all'utente se desidera inserire una parola, e chiederà di riprovare con un ciclo while finché // non viene inserito "si" o "no" dall'utente.

// Se l'utente dice "si", chiede di inserire la parola, eventuali sinonimi (fino a un massimo di 5, interrompendo con // "STOP"), e il significato della parola, dopo aver cercato spazio disponibile in indice[26].parole[30]

// (perché c'è il massimo di 30 parole esclusi sinonimi per lettera), la parola poi viene posizionata nel primo

// spazio disponibile della lettera corrispondente nel dizionario, se non c'è spazio disponibile per la lettera,

// informa l'utente e annulla l'inserimento, in caso è anche possibile non inserire sinonimi non rispondendo con // "si".

// Se l'utente dice "no", l'inserimento viene annullato.

// Al termine, richiama la funzione operazione(scelta(), indice) per continuare con un'altra scelta.

// Parametri in INPUT: dizionario indice[] (array di 26 elementi di tipo struct dizionario).

// Parametri in OUTPUT: nessuno.

**void** ins\_parola(dizionario indice[]){

**char** risposta[2];

**char** new\_parola[20];

**char** spiegazione[50];

**int** i, j, k;

printf("Desideri inserire una parola nel dizionario? (si/no): ");

scanf("%s", risposta);

strupr(risposta);

**while**(strcmp(risposta, "SI") != 0 && strcmp(risposta, "NO") != 0){

printf("RIPROVARE: ");

scanf("%s", risposta);

strupr(risposta);

}

**if**(strcmp(risposta, "SI") == 0){

printf("INSERISCI PAROLA:\n");

scanf("%s", new\_parola);

new\_parola[0] = toupper(new\_parola[0]);

**for**(i=0; i<26; i++){

**if**(new\_parola[0] == indice[i].lettera)

**break**;

}

// Cerco un posto disponibile per la parola

**for**(j = 0; j < 30; j++) {

**if**(indice[i].parole[j].parola\_sinonimi[0][0] == '\0') {

strcpy(indice[i].parole[j].parola\_sinonimi[0], new\_parola);

**break**;

}

}

**if**(j == 30) {

printf("Non c'è spazio per inserire altre parole per questa lettera.\n");

**return**;

}

printf("Vuoi inserire sinonimi? ");

**char** risp[2];

scanf("%s", risp);

strupr(risp);

**if**(strcmp(risp, "SI") == 0){

**for**(k=1; k<6; k++){

**char** new\_sinonimo[20];

printf("Sinonimo %d/5 (scrivere STOP per terminare): ",k);

scanf("%s", new\_sinonimo);

**if**(strcmp(new\_sinonimo, "STOP") == 0)

**break**;

**else**

strcpy(indice[i].parole[j].parola\_sinonimi[k], new\_sinonimo);

}

}

**else**

printf("Sinonimi non inseriti.\n");

printf("Inserisci significato: ");

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

fgets(spiegazione, sizeof(spiegazione), stdin);

strcpy(indice[i].parole[j].significato, spiegazione);

printf("Inserimento parola concluso.");

}

**Else**

printf("Inserimento parola annullato.");

operazione(scelta(), indice);

}

// Ricerca di una parola nel dizionario con eventuali sinonimi e significato.

// La funzione chiede all'utente se desidera cercare una parola e utilizza un ciclo while per richiedere "si" o "no" // finché l'input non è valido.

// Se l'utente dice "si", chiede di inserire la parola da cercare. La parola viene confrontata con le parole presenti // in indice[26].parole[30] (dove si possono inserire al massimo 30 parole per lettera).

// Se la parola viene trovata, la funzione stampa la parola, i suoi sinonimi (se presenti, fino a un massimo di 5) e il // significato.

// Se la parola non viene trovata, informa l'utente che non è presente nel dizionario.

// Se l'utente dice "no", la ricerca viene annullata.

// Al termine, richiama la funzione operazione(scelta(), indice) per continuare con un'altra scelta.

// Parametri in INPUT: dizionario indice[] (array di 26 elementi di tipo struct dizionario).

// Parametri in OUTPUT: nessuno.

**void** cerca\_parola(dizionario indice[]) {

**char** risposta[2];

**char** parola\_da\_cercare[20];

**int** i, j, k, trovato = 0;

printf("Desideri cercare una parola nel dizionario? (si/no): ");

scanf("%s", risposta);

strupr(risposta);

**while**(strcmp(risposta, "SI") != 0 && strcmp(risposta, "NO") != 0){

printf("RIPROVARE: ");

scanf("%s", risposta);

strupr(risposta);

}

**if**(strcmp(risposta, "SI") == 0){

printf("INSERISCI PAROLA DA CERCARE: ");

scanf("%s", parola\_da\_cercare);

parola\_da\_cercare[0]= toupper(parola\_da\_cercare[0]);

**for**(i= 0; i < 26; i++){

**if**(parola\_da\_cercare[0] == indice[i].lettera){

**for**(j= 0; j < 30; j++) {

**if**(strcmp(parola\_da\_cercare, indice[i].parole[j].parola\_sinonimi[0]) == 0){

trovato = 1;

printf("Parola trovata: %s\n", parola\_da\_cercare);

printf("Sinonimi:\n");

**for**(k= 1; k<6; k++) {

**if**(indice[i].parole[j].parola\_sinonimi[k][0] != '\0'){

printf("%s\n", indice[i].parole[j].parola\_sinonimi[k]);

}

}

printf("Significato: %s\n", indice[i].parole[j].significato);

**break**;

}

}

if(trovato)

**break**;

}

}

**if**(!trovato) {

printf("La parola %s non e' presente nel dizionario.\n", parola\_da\_cercare);

}

} **else**

printf("Ricerca parola annullata.\n");

operazione(scelta(), indice);

}

**TEST DI ESECUZIONE**

**1° TEST**

****

Il programma prima stamperà a schermo l’indice del dizionario, poi chiederà che operazione eseguire.  
In questo caso, ho inserito 1 (inserire una parola), il programma mi ha chiesto conferma, ed ho confermato digitando “si”.

Ho digitato e inserito la parola “Felicità”, il programma in seguito mi ha chiesto se volessi inserire sinonimi, per un massimo di 5 o finché non digitassi STOP.

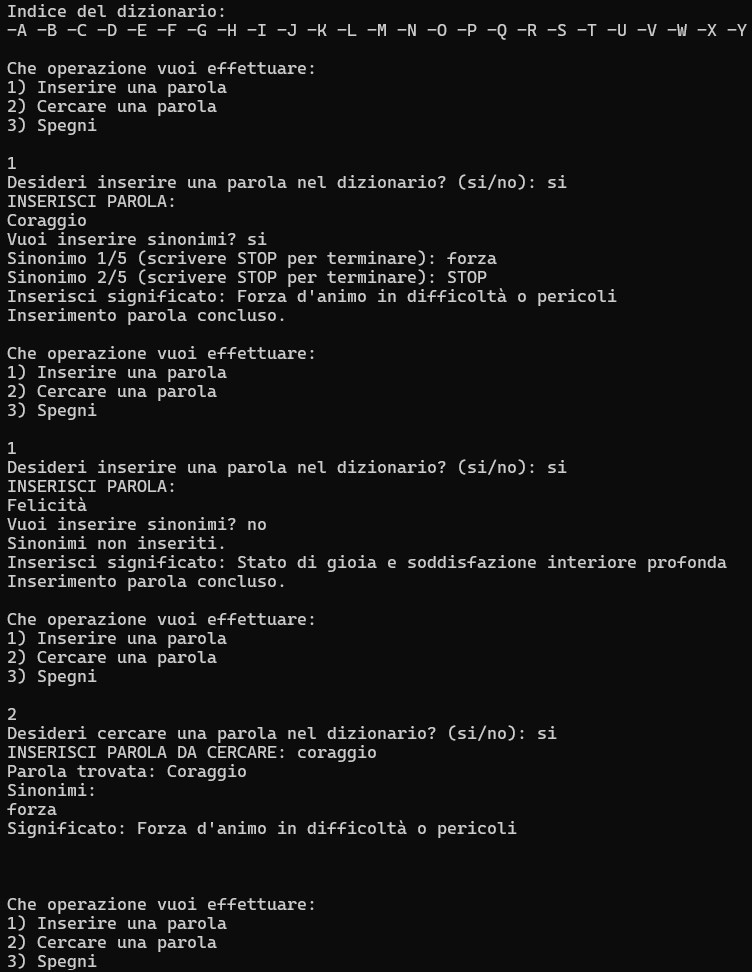
Una volta data la conferma ed inserito 5 sinonimi, il programma mi ha chiesto obbligatoriamente di inserire un significato alla parola, per poi concludere con “Inserimento parola concluso”.

In seguito, mi ha chiesto che altra operazione volessi eseguire (il programma lo chiederà in loop finché non si inserirà 3), ho inserito 2 (cercare una parola), e il programma mi ha chiesto conferma, ed ho confermato digitando “si”.

Cercando “felicità”, il programma cercherà la parola e mi stamperà a video la parola trovata, con i sinonimi inseriti e il suo significato.

E in loop, riinizierà la scelta della prossima operazione da eseguire.

Come notato, il programma è case insensitive.

**2° TEST**

Il programma prima stamperà a schermo l’indice del dizionario, poi chiederà che operazione eseguire.  
In questo caso, ho inserito 1 (inserire una parola), il programma mi ha chiesto conferma, ed ho confermato digitando “si”.

Ho digitato e inserito la parola “Coraggio”, il programma in seguito mi ha chiesto se volessi inserire sinonimi, per un massimo di 5 o finché non digitassi STOP.

Una volta data la conferma ed inserito un solo sinonimo, fermando la procedura con “STOP”, il programma mi ha chiesto obbligatoriamente di inserire un significato alla parola, per poi concludere con “Inserimento parola concluso”.

In seguito, mi ha chiesto che altra operazione volessi eseguire, ho eseguito di nuovo l’operazione 1, inserendo la parola “Felicità”, rifiutando con “no” la scelta di voler inserire sinonimi e aggiungendo infine il significato obbligatorio.

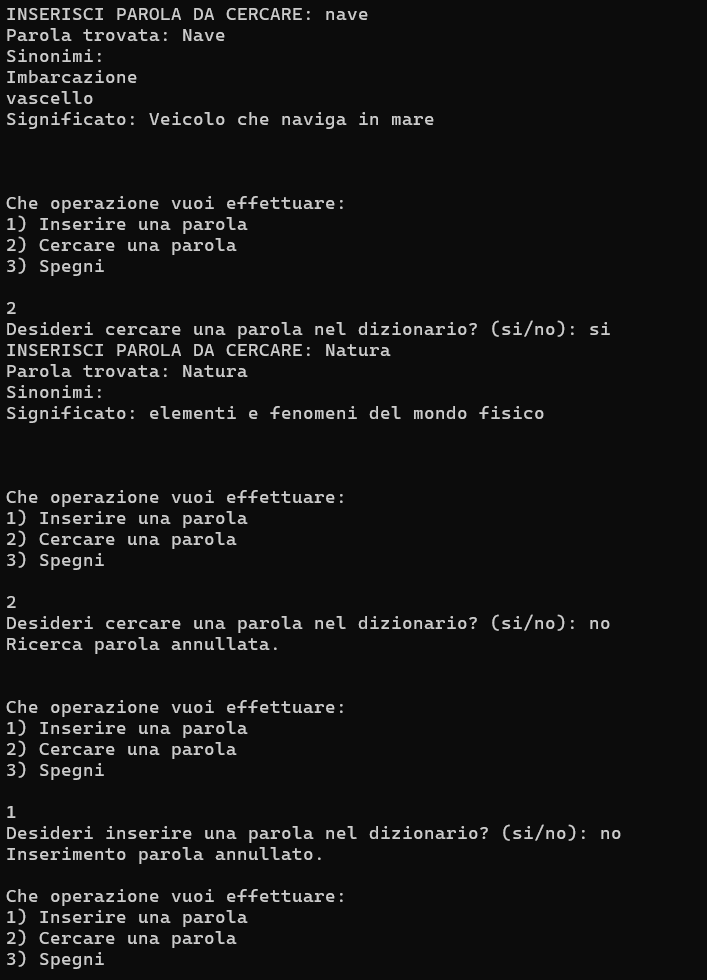
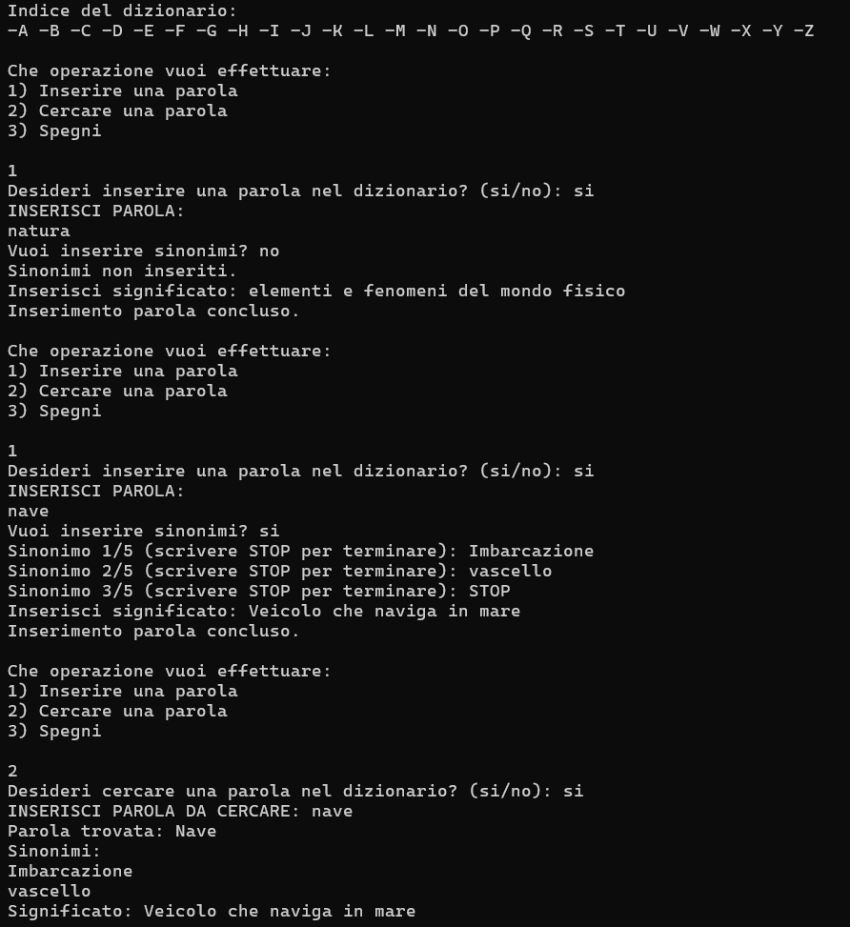
In seguito, ho eseguito l’operazione 2, dato conferma, e cercato la parola “coraggio”.

Il programma ha trovato la parola, dimostrando che prima non è stata sovrascritta dalla seconda parola inserita (Felicità), e stampando il sinonimo precedentemente inserito e il significato.

E in loop, riinizierà la scelta della prossima operazione da eseguire.

Come notato, il programma è case insensitive.

**3° TEST**

****

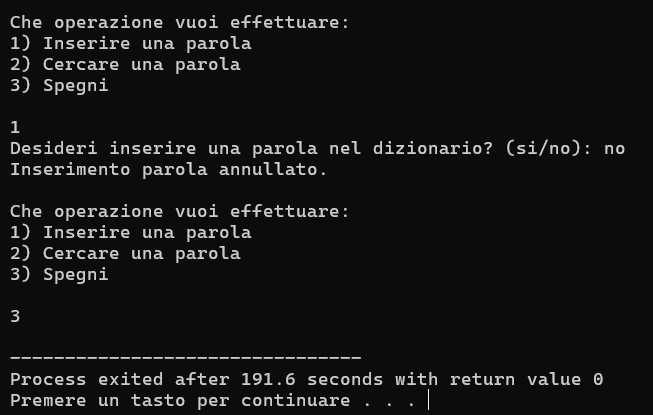
In questo test, ho eseguito l’operazione 1 (inserire una parola), per due volte consecutive, inserendo due parole differenti ma con stessa iniziale.

In seguito, ho eseguito per due volte l’operazione 2, ed ho cercato la prima volta la seconda parola precedentemente inserita, e poi ho cercato la prima parola precedentemente inserita.

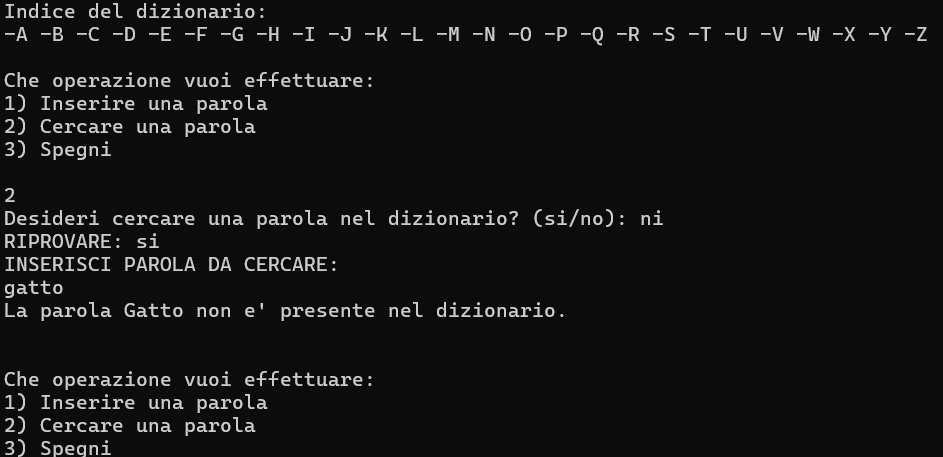
Il programma ha stampato correttamente entrambe le parole, con i corretti sinonimi e significati inseriti, dimostrando di aver salvato correttamente le due parole precedentemente inserite, senza nessuna sovrascrittura seppur avendo la stessa iniziale e stessa indicizzazione.

In seguito, si dimostra com’è possibile annullare operazioni di inserimento o ricerca, digitando “no” quando verrà chiesta conferma.

Come notato, il programma è case insensitive.

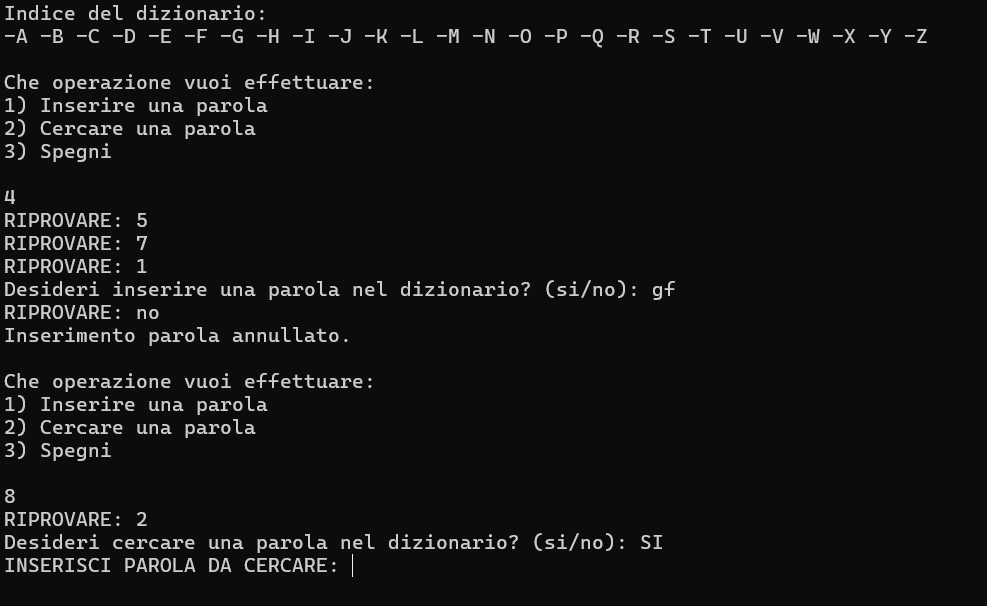
**4° TEST**

Il programma in questo caso mostra com’è possibile annullare un’operazione prima di svolgerla, digitando “no”, e come interrompere/spegnere il programma digitando “3”.

**5° TEST**

Il programma in questo caso mostra che digitando in modo errato “si” o “no” non verrà interrotta o eseguita nessuna operazione, ma verrà chiesto di riprovare a digitare e correttamente.

In seguito, mostra che non è possibile trovare nessuna parola se non è mai stata inserita prima.

**6° TEST**

Il programma in questo caso mostra che digitando in modo errato l’operazione che si vuole eseguire, quindi digitando numeri diversi da 1, 2 o 3, verrà chiesto di riprovare finché non verrà digitata un’operazione esistente ed eseguibile.

Mostrando infine che è case insensitive anche la conferma richiesta dopo che si sceglie un’operazione da eseguire, in questo caso il “SI”.