Politecnico di Milano
Informatica B, A.A. 2021/2022
Laboratorio 3

Luca Frittoli (luca.frittoli@polimi.it)
Mirko Salaris (mirko.salaris@polimi.it)

17 novembre 2021

1. **Statistiche di base** Si scriva un programma che prenda in input un array di numeri interi e stampi il minimo, massimo, media e somma.

Soluzione

```
#define N 20

#include <stdio.h>

int main()
{
    int vettore[N];
    int n; // lunghezza effettiva

    int minimo, massimo;
    // Salviamo la somma come float, per calcolarare la media esatta
    // Non ci servirà una variabile separata per la media
    float somma = 0;

    int i; // variabile ciclo

    printf("Quanti numeri si vogliono inserire? ");
    scanf("%d", &n);
    while (n <= 0 || n > N)
    {
        printf("Attenzione. Inserire un numero tra 1 e %d: ", N);
        scanf("%d", &n);
    }

    for (i = 0; i < n; i++)
    {
        printf("Numero i: ");
        scanf("%d", &vettore[i]);
    }

    // Troviamo minimo, massimo, somma e media in un unico ciclo
```

```
    minimo = vettore[0];
    massimo = vettore[0];

    for (i = 0; i < n; i++)
    {
        // Aggiorniamo il massimo
        if (vettore[i] > massimo)
            massimo = vettore[i];

        // Aggiorniamo il minimo
        if (vettore[i] < minimo)
            minimo = vettore[i];

        // Aggiorniamo la somma
        somma += vettore[i];
    }

    // %.0f stampa un float troncato a zero cifre decimali
    printf("Minimo:%d \t Massimo:%d \t Somma: %.0f \t Media %f",
           minimo, massimo, somma, somma / n);

    printf("\n");
    return 0;
}
```

2. **Aiuta gli escursionisti** Per aiutare degli escursionisti, scrivi un programma che accetti una sequenza di N dati di elevazione. È importante per gli escursionisti poter poi visualizzare la sequenza sia nell'ordine originale sia nella direzione opposta. Chiedi quindi in quale direzione si voglia visualizzare la sequenza e successivamente stampala a video, nella direzione richiesta.

   Bonus: il sentiero può essere percorso in entrambe le direzioni e partendo da qualsiasi punto. Riscrivi il programma per supportare questa funzionalità.

Soluzione

Versione base:

```
#include <stdio.h>

#define LUNGHEZZA_MASSIMA 100 // arbitrario

void main()
{
    float elev_sentiero[LUNGHEZZA_MASSIMA];
    int N;      // numero di elementi effettivo
    int i;      // per l'iterazione del ciclo
    int scelta; // 0 per "originale", 1 per "al contrario"

    // per gestire in modo comodo la stampa in entrambe le direzioni
    int inizio, incremento;
```

```
    printf("Quanti dati vuoi inserire?\n");
    scanf("%d", &N);

    // controllo dell'input
    while (N <= 0)
    {
        printf("Attenzione. Inserisci un numero maggiore di 0. \n");
        printf("Quanti numeri vuoi inserire?\n");
        scanf("%d", &N);
    }

    for (i = 0; i < N; i++)
    {
        // "i+1" perché per l'utente usiamo l'indicizzazione in base 1
        printf("Inserisci il dato di elevazione n.%d: ", i + 1);
        scanf("%f", &elev_sentiero[i]);
    }

    // nota: il primo numero è trattato diversamente, non avendo precedenti.
    printf("In che ordine vuoi visualizzare i dati del sentiero?\n");
    printf("Inserisci 0 per 'originale' e 1 per 'al contrario': ");
    scanf("%d", &scelta);
    while (scelta != 0 && scelta != 1)
    {
        printf("Attenzione. Inserisci 0 per 'originale' e 1 per 'al contrario': ");
        scanf("%d", &scelta);
    }

    if (scelta == 0)
    {
        inizio = 0;
        incremento = 1;
    }
    else
    {
        inizio = N - 1;
        incremento = -1;
    }

    // reset variabile i, la riutilizziamo per quest'altro ciclo
    i = inizio;
    printf("Dati elevazione sentiero: ");
    while (i >= 0 && i < N)
    {
        printf("%.2f ", elev_sentiero[i]);

        // effettivamente incrementata se incremento==+1, altrimenti decrementata
        i += incremento;
    }
    printf("\n");
}
```

3. **Raddoppi** Accetta dall'utente una sequenza di numeri interi positivi che termini con 0 e che sia lunga al massimo 100 numeri. Analizza quindi la sequenza e stampa le coppie di numeri consecutivi che sono uno il doppio del precedente.

   Esempio:

```
Sequenza di input: 1 2 7 14 3 6 12 10 11 0
Output:
    1 2
    7 14
    3 6
    6 12
```

Soluzione

Versione base:

```
#include <stdio.h>
#define LUNGHEZZA_MASSIMA 100

void main()
{
    int sequenza[LUNGHEZZA_MASSIMA];
    int N;    // lunghezza effettiva della sequenza
    int i;    // per l'iterazione del ciclo
    int stop; // variabile sentinella per rilevare lo zero

    int precedente, attuale; // variabili per analizzare il vettore

    stop = 0;
    N = 0;
    while (!stop && N < LUNGHEZZA_MASSIMA)
    {
        printf("Inserisci il %d° numero positivo della sequenza: ", N + 1);
        scanf("%d", &sequenza[N]);

        // controllo dell'input
        while (sequenza[N] < 0)
        {
            printf("Attenzione. "
                   "Inserisci un numero maggiore di 0 oppure 0 per terminare: ");
            scanf("%d", &sequenza[N]);
        }
        if (sequenza[N] == 0)
        {
            stop = 1;
        }
        else
        {
            N++;
        }
    }
```

```
    if (N > 0)
    {
        precedente = sequenza[0];
        for (i = 1; i < N; i++)
        {
            attuale = sequenza[i];
            if (attuale == precedente * 2)
            {
                printf("%d %d\n", precedente, attuale);
            }
            precedente = attuale;
        }
    }
    else
    {
        printf("La sequenza è vuota.");
    }

    printf("\n");
}
```

4. **Operazioni con matrici** Si scriva un programma che prenda in input 2 matrici di interi di dimensione 3×3, ne calcoli la somma (elemento per elemento) e la stampi a schermo.

   Esempio:

```
1 2 3     5 1 4      6  3  7
4 5 6  +  5 6 12  =  9  11 18
7 8 9     7 2 11     14 10 20
```

   Bonus: modificare il programma in modo che venga calcolato il prodotto riga per colonna fra matrici. Ricordarsi di verificare che le dimensioni delle matrici siano compatibili.

Soluzione

```
#include <stdio.h>

// Dimensione costante delle matrici
#define N 3
void main()
{
    // Definiamo le tre matrici
    int m1[N][N], m2[N][N], m3[N][N];
    // Indici dei cicli
    int i, j;
    // Prendiamo in input le due matrici
    // Per semplicita' lo facciamo con due soli cicli
    for (i = 0; i < N; i++)
    {
```

```
        for (j = 0; j < N; j++)
        {
            printf("Inserisci l'elemento (%d,%d) della prima matrice: ", i, j);
            scanf("%d", &m1[i][j]);
        }
    }

    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            printf("Inserisci l'elemento (%d,%d) della seconda matrice: ",
                   i, j);
            scanf("%d", &m2[i][j]);
            // Possiamo già calcolare la somma
            m3[i][j] = m1[i][j] + m2[i][j];
        }
    }

    printf("\nLa matrice somma e'\n");
    for (i = 0; i < N; i++)
    {
        for (j = 0; j < N; j++)
        {
            printf("%d\t", m3[i][j]);
        }
        printf("\n");
    }
}
```

5. **Area di un triangolo qualsiasi** Si scriva un programma che prenda in input le coordinate di tre punti nel piano e calcoli l'area del triangolo che abbia i tre punti come vertici.

   Consigli:

   - Salvare le coordinate dei vertici in una matrice 2×3 e le lunghezze dei lati in un array permette di evitare di ripetere parti di codice.
   - La formula di Erone permette di calcolare l'area di un triangolo date le lunghezze dei lati a, b, c e il semiperimetro p come $A = \sqrt{p(p-a)(p-b)(p-c)}$.
   - Controllare la correttezza del risultato ottenuto con triangoli rettangoli con lati paralleli agli assi.

Soluzione

```
#define D 2

#include <math.h>
#include <stdio.h>

void main()
```

```
{
    float vertici[D][3];
    float lati[3]; // lunghezze dei tre lati
    float p;       // semiperimetro
    float A;       // area
    int i, j, k;

    // Prendiamo in input i tre vertici
    for (j = 0; j < 3; j++)
    {
        for (i = 0; i < D; i++)
        {
            printf("Inserisci la coordinata %d del vertice %d: ", i, j);
            scanf("%f", &vertici[i][j]);
        }
    }

    // Calcoliamo le lunghezze dei lati (al quadrato)
    for (j = 0; j < 3; j++)
    {
        lati[j] = 0;
    }

    for (i = 0; i < D; i++)
    {
        for (j = 0; j < 3; j++)
        {
            lati[j] += (vertici[i][(j + 1) % 3] - vertici[i][(j + 2) % 3]) *
                       (vertici[i][(j + 1) % 3] - vertici[i][(j + 2) % 3]);
        }
    }

    // calcoliamo lati e semiperimetro
    p = 0;
    for (j = 0; j < 3; j++)
    {
        lati[j] = sqrt(lati[j]);
        p += lati[j] / 2;
    }

    // Calcoliamo l'area
    A = p;
    for (j = 0; j < 3; j++)
    {
        A *= p - lati[j];
    }

    A = sqrt(A);
    printf("L'area del triangolo e': %f\n", A);
}
```

6. **A - Una semplice struttura dati** Dichiarare un tipo di dato Esame che contenga il nome del corso e il voto ottenuto nel corso. Scrivere poi un programma che prenda in input un Esame e ne stampi i dati.

Soluzione

```
#include <stdio.h>
#define MAX_LEN 100

typedef struct
{
    char nome_corso[MAX_LEN];
    float voto_esame;
} Esame;

int main()
{
    Esame MioEsame;

    printf("Inserire un Esame\n");
    printf("Nome Corso = ");
    gets(MioEsame.nome_corso);

    printf("Voto = ");
    scanf("%f", &MioEsame.voto_esame);

    printf("\nL'esame letto e': %s con voto %f \n", MioEsame.nome_corso,
           MioEsame.voto_esame);
    return 0;
}
```

6. **B - Media dei voti** Dichiarare una struttura dati che usi il tipo di dati Esame definito nell'esercizio precedente per descrivere uno Studente tramite il nome e dalla lista degli esami sostenuti (3 esami). Scrivere poi un programma che prenda in input uno studente e calcoli la media dei suoi voti.

Soluzione

```
#include <stdio.h>
#define MAX_LEN 100
#define N 3
typedef struct
{
    char nome_corso[MAX_LEN];
    float voto_esame;
} Esame;

typedef struct
{
    char nome_studente[MAX_LEN];
    Esame voti[N];
```

```
} Studente;

int main()
{
    Studente st;
    int i;
    float media = 0;
    printf("Inserire lo studente\n");
    printf("Nome Studente = ");
    gets(st.nome_studente);
    for (i = 0; i < N; i++)
    {
        printf("Inserire Esame %d\n", i + 1);
        printf("Nome Corso = ");
        gets(st.voti[i].nome_corso);
        printf("Voto = ");
        scanf("%f", &st.voti[i].voto_esame);
        fflush(stdin);
        printf("\nL'esame letto e': %s con voto %f \n",
               st.voti[i].nome_corso, st.voti[i].voto_esame);
        media += st.voti[i].voto_esame;
    }
    printf("La media dei voti e': %f\n", media / N);
    return 0;
}
```

7. **Tema d'Esame del 25/01/2021 (10 punti)** Giovo vi ha incaricato di creare un programma per la gestione delle consegne a domicilio. Il programma mantiene un vettore di richieste di consegne a domicilio prodotte dai vari clienti (si chiami questa variabile: `richieste`); ciascuna richiesta è caratterizzata dal codice cliente (`codice_cliente`, un intero), indirizzo del cliente (`ind_cli`, codificato per semplicità in termini di due coordinate GPS: `float x` e `y`), indirizzo del commerciante (`ind_com`, anch'esso codificato come due coordinate GPS: `float x` e `y`) e prezzo pagato per la consegna (`costo_merce`). Il programma mantiene inoltre un vettore di fattorini (nome variabile: `fattorini`), caratterizzati ciascuno dal codice fattorino (`codice_fattorino`, un intero), dalla posizione (sempre in termini di due coordinate GPS: `float x` e `y`) e da una variabile logica occupato che indica se il fattorino sta già effettuando una consegna oppure no.

   1. Si definiscano in linguaggio C i tipi necessari a rappresentare richieste e fattorini. Si dichiarino, inoltre, le variabili array `richieste` e `fattorini` dimensionandole tramite due costanti definite con nome `N_RICHIESTE` e `N_FATTORINI`, rispettivamente.
   2. Si scriva una porzione di codice in linguaggio C che, partendo dai due vettori `fattorini` e `richieste` completamente popolati, per ogni richiesta, ricavi l'indice del fattorino libero più vicino al commerciante (a tale scopo, si calcoli la distanza Euclidea tra fattorino e commerciante). La porzione di codice applica, inoltre, uno sconto del 20% al costo merce se la distanza tra commerciante e fattorino è inferiore alla distanza fra commerciante e cliente (si modifichi il campo `costo_merce` della richiesta per applicare lo sconto). Per ogni richiesta, si stampino l'indice della richiesta, le coordinate del cliente, del commerciante e del fattorino, insieme al costo eventualmente scontato.

   Suggerimento: La distanza euclidea tra due punti in (x_1, y_1) e (x_2, y_2) è definita nel seguente modo: $d = \sqrt{(x_1 - x_1)^2 + (y_1 - y_2)^2}$.

Soluzione

```
#include <stdio.h>
// prima parte
#define N_RICHIESTE 10
#define N_FATTORINI 5
typedef struct
{
    int codice;
    float ind_cli[2];
    float ind_com[2];
    float costo_merce;
} Ric;
typedef struct
{
    int codice;
    float pos[2];
    int libero;
} Fat;

void main()
{
    Ric richieste[N_RICHIESTE];
    Fat fattorini[N_FATTORINI];

    // seconda parte
    float dist, min_dist, dist_cli;
    int idx = -1;
    for (int i = 0; i < N_RICHIESTE; i++)
    {
        for (int j = 0; j < N_FATTORINI; j++)
        {
            if (fattorini[j].libero && idx == -1)
            {
                // calcolo la distanza del primo fattorino libero
                min_dist = (richieste[i].ind_com[0] - fattorini[j].pos[0]) *
                            (richieste[i].ind_com[0] - fattorini[j].pos[0]) +
                           (richieste[i].ind_com[1] - fattorini[j].pos[1]) *
                            (richieste[i].ind_com[1] - fattorini[j].pos[1]);
                idx = j;
            }
            else if (fattorini[j].libero && idx != -1)
            {
                dist = (richieste[i].ind_com[0] - fattorini[j].pos[0]) *
                           (richieste[i].ind_com[0] - fattorini[j].pos[0]) +
                       (richieste[i].ind_com[1] - fattorini[j].pos[1]) *
                           (richieste[i].ind_com[1] - fattorini[j].pos[1]);
                if (dist < min_dist)
                {
                    // controllo se c'è un fattorino più vicino
                    min_dist = dist;
```

```
                    idx = j;
                }
            }
        }
        if (idx == -1)
        {
            printf("Non ci sono fattorini liberi\n");
        }
        else
        {
            // calcolo la distanza dal cliente
            dist_cli = (richieste[i].ind_com[0] - richieste[i].ind_cli[0]) *
                        (richieste[i].ind_com[0] - richieste[i].ind_cli[0]) +
                    (richieste[i].ind_com[1] - richieste[i].ind_cli[1]) *
                        (richieste[i].ind_com[1] - richieste[i].ind_cli[1]);
            if (min_dist < dist_cli) // in questo caso applichiamo lo sconto
            {
                richieste[i].costo_merce *= 0.8;
            }
            // stampiamo tutto
            printf("Richiesta %d\n", richieste[i].codice);
            printf("Coordinate cliente: (%f, %f)\n", richieste[i].ind_cli[0],
                   richieste[i].ind_cli[1]);
            printf("Coordinate commerciante: (%f, %f)\n",
                   richieste[i].ind_com[0], richieste[i].ind_com[1]);
            printf("Coordinate fattorino: (%f, %f)\n", fattorini[idx].pos[0],
                   fattorini[idx].pos[1]);
            printf("Costo: %f\n\n", richieste[i].costo_merce);
        }
    }
}
```

8. **Autosalone** Definire una struttura che descriva un Modello di automobile in termini di marchio, nome del modello e cilindrata (per esempio `marchio = 'Fiat'`, `nome = 'Panda'`, `cilindrata = 1000`), e una struttura che descriva un'Automobile in termini di Modello, cognome del proprietario e anno di immatricolazione. Scrivere poi un programma che permetta all'utente di inserire i dati di un numero a scelta di automobili di un autosalone (compreso tra 1 e 10) e stampi a video:

    1. il numero di automobili di un marchio (scelto dall'utente) immatricolati a partire dal 2019, e
    2. i cognomi dei proprietari di automobili di cilindrata superiore a 1800.

Soluzione

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

#define MAX 10

typedef struct
```

```
{
    char marchio[20];
    char nome[20];
    int cilindrata;
} Modello;

typedef struct
{
    Modello modello;
    char proprietario[30];
    int anno;
} Automobile;

int main(void)
{
    int i, n, anno, tot = 0, condition;
    Automobile autosalone[MAX];
    char marchio[20];

    do
    {
        printf("Quante auto vuoi inserire?: ");
        scanf("%d", &n);
    } while (n < 1 || n > MAX);

    for (i = 0; i < n; i++)
    {
        printf("\nAutomobile %d", i + 1);
        Modello modello;
        // acquisiamo i dati del modello
        printf("\nMarchio: ");
        scanf("%s", modello.marchio);
        printf("Modello: ");
        scanf("%s", modello.nome);
        printf("Cilindrata: ");
        scanf("%d", &modello.cilindrata);
        autosalone[i].modello = modello;
        // acquisiamo i dati su proprietario e anno
        printf("Proprietario: ");
        scanf("%s", autosalone[i].proprietario);
        printf("Anno: ");
        scanf("%d", &autosalone[i].anno);
    }
    // facciamo scegliere all'utente un marchio
    printf("\nQuale marchio di auto cerchi?: ");
    scanf("%s", marchio);
    printf("%s", marchio);
    // contiamo quante auto sono del marchio scelto
    for (i = 0; i < n; i++)
    {
        condition = strcmp(autosalone[i].modello.marchio, marchio);
```

```
        if (condition == 0 && autosalone[i].anno >= 2019)
            tot++;
    }
    printf("\n\n%d auto di marchio %s sono state immatricolate dal 2019",
           tot, marchio);
    // stampiamo i nomi dei proprietari di auto di grossa cilindrata
    printf("\n\nProprietari con auto di cilindrata maggiore di 1800 cc:\n");
    for (i = 0; i < n; i++)
        if (autosalone[i].modello.cilindrata >= 1800)
            printf(" %s\n", autosalone[i].proprietario);
    return 0;
}
```

9. **Maiuscole/minuscole** Si scriva un programma che prenda in input una stringa e converta i caratteri minuscoli in maiuscoli e vice-versa.
Nota: per acquisire una stringa di testo utilizza la funzione `fgets(variabile_stringa,max_caratteri, stdin)`, dove `stdin` va scritto esattamente così.
Esempio:

```
char testo[240];
fgets(testo, 240, stdin);
```

Soluzione

```
#include <stdio.h>
#include <string.h>
#define MAX_LEN 100
int main()
{
    // Definiamo la stringa
    char s[MAX_LEN];
    int len, i;
    printf("Inserisci una stringa: ");
    fgets(s, MAX_LEN, stdin);
    // Calcolo la lunghezza della stringa
    len = strlen(s);
    // Cerchiamo le vocali
    for (i = 0; i < len; i++)
    {
        //Convertiamo in carattere maiuscolo aggiungengo a codice ascii la
        //differenza tra un carattere maiuscolo e uno minuscolo
        if (s[i] >= 'a' && s[i] <= 'z')
            s[i] += 'A' - 'a';
        //Convertiamo in carattere minuscolo aggiungengo al codice ascii la
        //differenza tra un carattere minuscolo e uno maiuscolo
        else if (s[i] >= 'A' && s[i] <= 'Z')
            s[i] += 'a' - 'A';
    }
    printf("La stringa risultate e :\n");
```

```
    printf("%s", s);
    return 0;
}
```

10. **Suddivisione in parole** Nelle applicazioni di NLP (processazione del linguaggio "umano") spesso è richiesto lavorare sulle singole parole. Come esercizio, scrivi quindi un programma in grado di separare le singole parole all'interno di una frase. Il tuo programma deve accettare in input una stringa di testo di qualsiasi lunghezza e restituire in output la lista delle parole in essa contenuta, andando a capo ad ogni parola.

    Consiglio: il programma completo dovrebbe gestire anche la punteggiatura (eliminandola), però prova ad iniziare considerando testo composto solo da lettere e spazi.

Soluzione

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

#define LUNGHEZZA_MASSIMA 240 // arbitrario

void main()
{
    char input[LUNGHEZZA_MASSIMA];
    int N; // lunghezza effettiva stringa
    int i; // per l'iterazione del ciclo

    // se a_capo==1, dall'ultima volta che siamo andati a capo non abbiamo ancora
    // stampato nessun carattere di testo.
    // Questo ci permette di andare a capo una volta sola, indipendetemente
    // da quanta punteggiatura ci sia.
    int a_capo;

    printf("Inserisci del testo da separare in parole\n");

    fgets(input, LUNGHEZZA_MASSIMA, stdin);

    N = strlen(input);
    a_capo = 0;
    printf("Lista parole: \n");
    for (i = 0; i < N; i++)
    {
        if ((input[i] >= 'a' && input[i] <= 'z') ||
            (input[i] >= 'A' && input[i] <= 'Z'))
        {
            printf("%c", input[i]);
            a_capo = 0;
        }
        else
        {
            if (!a_capo)
```

```
                printf("\n");
            // else... ma è inutile specificarlo.
            a_capo = 1;
        }
    }
}
```

Esercizi extra

Extra 1. **Parentesi** Come esercizio introduttivo alla programmazione di una calcolatrice scientifica, scrivi un programma che accetti in input una stringa contenente parentesi tonde e verifichi che il numero di parentesi aperte e chiuse combaci.

Esempi:

```
()()    -> Sì
(())    -> Sì
)))(((  -> Sì
())     -> No
```

Bonus: restituisci un responso positivo solo se le parentesi sono correttamente innestate.

Esempi:

```
()()    -> Sì
(())    -> Sì
)))(((  -> No
())     -> No
```

Soluzione

Versione base

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

#define LUNGHEZZA_MASSIMA 100 // arbitrario

void main()
{
    char input[LUNGHEZZA_MASSIMA];
    int N; // lunghezza effettiva stringa
    int i; // per l'iterazione del ciclo
    int numero_aperte, numero_chiuse;
    int errore; // variabile sentinella

    do
    {
        printf("Inserisci una stringa composta da parentesi '(' e ')': \n");
        fgets(input, LUNGHEZZA_MASSIMA, stdin);
```

```
        // fgets legge anche \n in fondo. Rimuoviamo un carattere
        N = strlen(input) - 1;
        errore = 0;
        numero_aperte = 0;
        numero_chiuse = 0;
        for (i = 0; i < N && !errore; i++)
        {
            if (input[i] == '('
            {
                numero_aperte++;
            }
            else if (input[i] == ')')
            {
                numero_chiuse++;
            }
            else
            {
                errore = 1;
                printf("Attenzione, la stringa deve essere composta solo "
                "da parentesi tonde aperte e chiuse: '(', ')'.\n\n");
            }
        }
    } while (errore);

    if (numero_aperte==numero_chiuse)
    {
        printf("Bravo, il numero di parentesi aperte e chiuse combaciano.");
    }
    else
    {
        printf("Mi spiace, il numero di parentesi aperte e chiuse non combaciano.");
    }
    printf("\n");
}
```

Extra 2. **Cifrario di Cesare** Si scriva un programma che prenda in input una stringa (plaintext) senza spazi e fatta solo di caratteri maiuscoli, e un numero intero (key). Il programma cifra la stringa plaintext con il Cifrario di Cesare con chiave key, stampando il risultato cifrato (cyphertext) a schermo.
Il Cifrario di Cesare è un semplice cifrario a rotazione, che sostituisce una lettera dell'alfabeto con un'altra, ruotandole di key posti.

Esempio:

```
key = 3
plaintext  = A B C D E F G ... X Y Z
cyphertext = D E F G H I J ... A B C
```

Bonus: Scambia un messaggio cifrato, e la chiave, con un tuo collega e provate a decifrarlo (suggerimento: il programma accetta anche chiavi negative).

Soluzione

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

#define MAX_LEN 100

int main()
{
    char plaintext[MAX_LEN], cyphertext[MAX_LEN];
    int key;
    int i, lunghezza;

    printf("Inserisci il messaggio da cifrare: ");
    // Uso scanf supponendo che la stringa non contenga spazi
    // Supponiamo inoltre che la stringa inserita contenga solo caratteri maiuscoli
    scanf("%s", plaintext);
    printf("Inserisci la chiave: ");
    scanf("%d", &key);

    lunghezza = strlen(plaintext);
    // Cifro ogni carattere del plaintext
    for (i = 0; i < lunghezza; i++)
    {
        // L'idea della seguente istruzione e':
        // - Traslo il carattere i-esimo tra 0 e 25 (sottraendo 'A')
        // - Gli sommo la chiave
        // - Faccio modulo 26 -> il risultato e' ancora tra 0 e 25
        // - Riporto il tutto nell'intervallo originale sommando 'A'
        // Esempio: se inserisco Z e chiave 1 ho:
        //   Z - A = 25
        //   Z - A + 1 = 26
        //   (Z - A + 1) % 26 = 0
        //   (Z - A + 1) % 26 + 'A' = A
        cyphertext[i] = ((plaintext[i] + key - 'A') % 26) + 'A';
    }
    // Aggiungo il terminatore
    cyphertext[i] = '\0';
    printf("Il messaggio cifrato e': %s\n", cyphertext);
    return 0;
}
```