Politecnico di Milano
Informatica B, AA 2020/2021
Laboratorio 2

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

15 Ottobre 2020

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 1: ");
        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. **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;
    }
```

4. **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;
        }
    }

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

5. **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");
}
```

6. **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");
}
```

7. **Somma di vettori** Si scriva un programma che prenda in input 2 array (della stessa lunghezza) di numeri reali, ne calcoli la somma (elemento per elemento) e successivamente la stampi a schermo.

   Bonus: Modifica il programma per calcolare il prodotto scalare tra due vettori.

Soluzione

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

```
// Lunghezza massima array
#define N 20
int main()
{
    int vettore1[N], vettore2[N], somma[N];
    int n; // lunghezza effettiva
    int i; // Indice ciclo

    printf("Quanto sono lunghi i due vettori?");
    scanf("%d", &n);

    while (n <= 0 || n > N)
    {
        printf("Inserisci un numero maggiore di 0 e minore di %d: ", N);
        scanf("%d", &n);
    }

    // Prendiamo in input i due vettori
    for (i = 0; i < n; i++)
    {
            printf("Inserisci l'elemento %d del primo vettore: ", i);
            scanf("%d", &vettore1[i]);
    }
    for (i = 0; i < n; i++)
    {
        printf("Inserisci l'elemento %d del secondo vettore: ", i);
        scanf("%d", &vettore2[i]);
    }


    for (i = 0; i < N; i++)
    {
        somma[i] = vettore1[i] + vettore2[i];
    }

    printf("\nIl vettore somma e': ");
    printf("[");
    for (i = 0; i < n-1; i++)
    {
        printf("%d, ", somma[i]);
    }
    // l'ultimo elemento lo stampiamo fuori dal ciclo solo per gestire
    // la parentesi quadra
    printf("%d]", somma[n-1]);

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

8. **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:

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

int main(void)
{
    //le coordinate dei vertici sono degli array
    float v1[D], v2[D], v3[D];
    float a, b, c; // lunghezze dei tre lati
    float p; // semiperimetro
    float A; // area
    int i;

    // Prendiamo in input i due vettori
    for (i = 0; i < D; i++)
    {
        printf("Inserisci la coordinata %d del primo vertice: ", i);
        scanf("%f", &v1[i]);
    }
    for (i = 0; i < D; i++)
    {
        printf("Inserisci la coordinata %d del secondo vertice: ", i);
        scanf("%f", &v2[i]);
    }
    for (i = 0; i < D; i++)
    {
        printf("Inserisci la coordinata %d del terzo vertice: ", i);
        scanf("%f", &v3[i]);
    }

    // Calcoliamo le lunghezze dei lati e il semiperimetro
    a = 0;
    b = 0;
    c = 0;
    p = 0;
    for (i = 0; i < D; i++)
    {
        a += (v2[i] - v1[i]) * (v2[i] - v1[i]);
        b += (v3[i] - v2[i]) * (v3[i] - v2[i]);
        c += (v1[i] - v3[i]) * (v1[i] - v3[i]);
    }
```

```
    a = sqrt(a);
    b = sqrt(b);
    c = sqrt(c);
    p = (a + b + c) / 2;

    // Calcoliamo l'area
    A = sqrt(p * (p - a) * (p - b) * (p - c));
    printf("L'area del triangolo e': %f\n", A);
    return 0;
}
```

9. **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;
}
```