Testo della prova

Si dichiarino due vettori VET1 e VET2 di numeri reali, di cardinalità N. Gli elementi di tali vettori siano inseriti da tastiera dall'utente. Si progetti una funzione che restituisca come parametro di uscita il prodotto scalare tra i due vettori.

Si ricorda che in geometria analitica, dati due vettori $\boldsymbol{a} = [a_1, a_2, \ldots, a_n]$ e $\boldsymbol{b} = [b_1, b_2, \ldots, b_n]$, di definisce prodotto scalare: $\boldsymbol{a} \times \boldsymbol{b} = a_1b_1 + a_2b_2 + \ldots + a_nb_n$.

Esempio

INPUT:

VET1 = [20.7, 23.4, 22.0, 21.2, 25.3, 23.2, 24.1]

VET2 = [12.3, 11.1, 25.6, 24.4, 10.0, 21.1, 17.6]

OUTPUT:

Il prodotto scalare è: 2761.51

Svolgimento

Data la definizione di prodotto scalare tra due vettori come somma dei prodotti degli elementi omologhi, e considerato che gli elementi dei due vettori sono numeri reali, è chiaro che il risultato dell'operazione sarà a sua volta un numero reale. Pertanto, la funzione "ProdScalare" che andremo a realizzare dovrà avere un prototipo di questo tipo:

```
float ProdScalare(const float v1[], const float v2[], int riemp);
```

Poiché in C++ I vettori sono sempre scambiati per riferimento, è opportuno forzare la non modificabilità dei parametri di ingresso v1 e v2 aggiungendo la clausola **const**. Inoltre, poiché la funzione non ha alcun modo per recuperare autonomamente il riempimento dei vettori, cioè il numero effettivo di elementi che li compongono, è necessario passarle anche questa informazione.

L'implementazione della funzione dovrà prevedere una variabile in cui accumulare la somma dei vari prodotti degli elementi omologhi. Chiamiamo "res" tale variabile, che inizializziamo a zero in quanto andremo di volta in volta a sommare i prodotti che calcoliamo al suo valore attuale:

```
float res = 0;
```

Ricordiamo che l'ultimo elemento utile dei vettori corrisponde all'indice **riemp -1**, in quanto gli indici partono da zero. Pertanto, l'obiettivo è calcolare:

```
v1[0] * v2[0]

v1[1] * v2[1]

...

v1[riemp-1] * v2[riemp-1]
```

Per operare sugli elementi dei vettori, è opportuno impostare un ciclo for:

```
for(int i=0; i<riemp; i++) {

...

}
```

Ad ogni iterazione del ciclo, possiamo calcolare il prodotto **v1[i] * v2[i]** e salvarlo in una variabile temporanea che chiameremo **tmp**:

```
float tmp = v1[i] * v2[i];
```

Il valore appena calcolato (e contenuto nella variabile **tmp**) può essere sommato al valore attuale di **res**:

```
res = res + tmp;
```

In alternativa, è possibile usare l'operatore **+=**, che effettua in sequenza una somma ed una assegnazione:

```
res += tmp;  // è equivalente a scrivere res = res + tmp;
```

In definitiva, il ciclo for che permette di calcolare tutti i prodotti degli elementi omologhi e sommarli tra loro è il seguente:

```
for(int i=0; i<riemp; i++) {
      float tmp = v1[i] * v2[i];
      res += tmp;
}
```

Al termine del ciclo, la variabile res conterrà la somma di tutti i prodotti degli elementi omologhi, che può quindi essere restituita al programma chiamante mediante l'istruzione **`return`**.