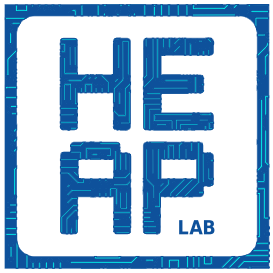
Esercizi sui vettori in C

Stefano Cherubin*



17/10/2019

[**Informatica A**] Esercitazione #6

corso per Ing. Gestionale a.a. 2019/20

*<nome>.<cognome>@polimi.it

Indice

1 Analisi del fatturato

Sono arrivati dal reparto vendite i dati relativi all'andamento del volume di mercato dell'ultimo anno. I dati rappresentano le quantità vendute per ogni mese dell'anno. Si calcoli il mese con il massimo volume di vendita e la più lunga serie di vendite strettamente crescente.

1.1 Approccio alla soluzione

Il problema richiede la ricerca del valore massimo e della più lunga sequenza di valori crescenti.

È possibile calcolare questi parametri contemporaneamente all'inserimento dei dati, aggiornando variabili temporanee tra un dato e il seguente. Un approccio alternativo consiste nel memorizzare i dati in un vettore e scansire il vettore in un secondo momento. A scopo didattico, si mostrerà il secondo approccio nella soluzione che segue.

Non sapendo le unità di misura del volume, si può fare una assunzione esplicita che si tratti di unità vendute e quindi utilizzare il tipo **int** oppure utilizzare il tipo **float** senza imporre vincoli o assunzioni sulle unità di misura dei dati in ingresso. In entrambi i casi, trattandosi di quantità nette vendute, si dovrà verificare che i dati in ingresso siano positivi o nulli.

1.2 Due parametri, un algoritmo

Il problema richiede di estrarre due parametri dal vettore. Per separare semanticamente i due algoritmi che si devono implementare sarebbe consigliato di ricavare un parametro senza modificare i dati di input e poi eseguire in seguito il secondo algoritmo per ricavare il secondo parametro.

```
for (i = 0; i < N; ++i) {
  // find max
}
for (i = 0; i < N; ++i) {
  // find sequence
}
```

I due algoritmi sono molto simili come struttura, eseguono il medesimo ciclo di scansione del vettore. Per ragioni di efficienza è sempre auspicabile ridurre al minimo il numero di scansioni di un vettore. Di seguito si presenta una soluzione che racchiude in un unico algoritmo l'estrazione di entrambi i parametri.

```
for (i = 0; i < N; ++i) {
  // find max
  // find sequence
}
```

1.3 Posizione o valore

Quando si eseguono operazioni di ricerca (di un elemento, del massimo, del minimo, di un intervallo, ...) all'interno di un vettore si può scegliere se lavorare con il valore o con la posizione. Ad esempio durante la ricerca del massimo, si può memorizzare il valore massimo e aggiornarlo ogni volta.

```
for (i = 0; i < N; ++i) {
  if (v[i] > max) {
    max = v[i];
  }
}
```

Un altro approccio è quello di aggiornare la posizione del valore massimo; utilizzando la posizione, è sempre possibile tornare al valore massimo e consente di occupare in memoria solamente una variabile per l'indice relativo alla posizione dell'elemento di interesse. Inoltre risulta più semplice effettuare ordinamenti più complessi (ordinamento su più livelli, ordinamento di elementi strutturati, ...).

```
for (i = 0; i < N; ++i) {
  if (v[i] > v[pmax]) {
    pmax = i;
  }
}
```

1.4 Soluzione C - Store and scan

Listato 1: Analisi fatturato

```
1  #include <stdio.h>
2  #define INVALID_POS -1
3  #define L 12
4  int main () {
5    float volume[L];
6    int i, p1, p2, current_p1, current_p2, pmax;
7
8    /* inserimento dati - reparto vendite */
9    for (i = 0; i < L; ++i) {
10     printf("\nVolume del mese %d: ", i);
11     scanf("%f", &volume[i]);
12     if (volume[i] < 0) {
13       printf("\nVolumi negativi non ammessi");
14       --i;
15     }
16   }
17
18   /* inizializzazioni */
19   pmax = 0; /* posizione valore massimo */
20
21   /* posizioni della sequenza più lunga */
22   p1 = INVALID_POS;
23   /* p2 - p1 inizializzata negativa */
24   p2 = INVALID_POS -1;
25
26   /* posizioni della sequenza corrente */
27   current_p1 = INVALID_POS;
28   current_p2 = INVALID_POS - 1;
29
30   /* scansione vettore */
31   for (i = 1; i < L; ++i) {
32
33     /* ricerca del massimo */
34     if (volume[pmax] < volume[i]) {
35       pmax = i;
36     }
37
38     /* ricerca sequenza crescente */
39     if (volume[i] > volume[i - 1]) {
40       current_p2 = i;
41
```

```
42        /* aggiorna current_p1 solo all'inizio */
43        if (current_p1 == INVALID_POS) {
44          /* la sequenza inizia a i - 1  */
45          current_p1 = i - 1;
46        }
47
48        /* ricerca sequenza più lunga */
49        if ((current_p2 - current_p1) > (p2 - p1)) {
50          p2 = current_p2;
51          p1 = current_p1;
52        }
53      } else {
54        current_p1 = INVALID_POS;
55        current_p2 = INVALID_POS - 1;
56      }
57    }
58
59    printf("\nIl massimo si è avuto al mese %d con %f",
         pmax, volume[pmax]);
60    printf("\nLa più lunga sequenza positiva è dal mese
         %d al mese %d.", p1, p2);
61    return 0;
62  }
```

1.5 Variante: confronto con l'anno passato

> Si vuole confrontare i dati di quest'anno con i dati dell'anno scorso. Si acquisiscano due sequenze di volumi. Si assuma che questi volumi rappresentino unità di prodotto vendute e siano interi positivi (o nulli). Si determini in quale mese si è avuto l'incremento massimo rispetto al corrispettivo del precedente anno e si identifichi la più lunga sequenza di incrementi positivi.

In questa variante l'approccio alla risoluzione è il medesimo utilizzato in precedenza. Si introduce una parte di calcolo per ricavare il vettore degli incrementi, l'effettiva struttura dati su cui viene applicato l'algoritmo.

Listato 2: Analisi fatturato: confronto con l'anno

```
1  #include <stdio.h>
2  #define L 12
3  #define INVALID_POS -1
4  int main () {
5    float inc[L];
6    int volume[L];
7    int old[L];
8    int i, p1, p2, current_p1, current_p2, pmax;
9
10   for (i = 0; i < L; ++i) {
11     printf("\nVolume del mese %d per l'anno corrente:
         ", i);
12     scanf("%d", &volume[i]);
13     if (volume[i] < 0) {
14       printf("\nVolumi negativi non ammessi");
15       --i;
16     }
17   }
18   for (i = 0; i < L; ++i) {
19     printf("\nVolume del mese %d per l'anno passato: "
         , i);
20     scanf("%d", &old[i]);
21     if (old[i] < 0) {
22       printf("\nVolumi negativi non ammessi");
23       --i;
24     }
25   }
26
27   for (i = 0; i < L; ++i) {
28     if (old[i] != 0) { /* evita divisione per 0 */
29       inc[i] = (float) (volume[i] - old[i]) / old[i];
30     } else {
31       inc[i] = 9999.9;/* valore di comodo */
```

```
32      }
33    }
34
35    pmax = 0;
36    p1 = INVALID_POS;
37    p2 = INVALID_POS - 1;
38    current_p1 = INVALID_POS;
39    current_p2 = INVALID_POS - 1;
40
41    for (i = 0; i < L; ++i) {
42      if (inc[pmax] < inc[i]) {
43        pmax = i;
44      }
45      if (inc[i] > 0.0) {
46        current_p2 = i;
47        if (current_p1 == INVALID_POS) {
48          current_p1 = i;
49        }
50        if ((current_p2 - current_p1) > (p2 - p1)) {
51          p2 = current_p2;
52          p1 = current_p1;
53        }
54      } else {
55        current_p1 = INVALID_POS;
56        current_p2 = INVALID_POS - 1;
57      }
58    }
59
60    printf("\nIl massimo si è avuto al mese %d con %f",
         pmax, inc[pmax]);
61    if (p1 != INVALID_POS) {
62      printf("\nLa più lunga sequenza positiva è dal
           mese %d al mese %d.", p1, p2);
63    } else {
64      printf("\nNon esiste una sequenza positiva
           rispetto al precedente anno");
65    }
66    return 0;
67  }
```

2 Merge

Scrivere un programma in linguaggio C che lette dallo standard input due sequenze vettoriali ordinate di interi V1[n], V2[m] ne crei una terza V3[n+m] anch'essa ordinata, che contenga tutti gli elementi di V1 e di V2 .

2.1 Soluzione C

Sfruttando il fatto che le sequenze V1 e V2 sono già ordinate, posso scandirle progressivamente entrambe e per ogni elemento da inserire in V3 scegliere l'elemento minore tra le due sorgenti. Quando una delle due sorgenti termina, esaurisco gli elementi residui copiandoli in coda al risultato V3.

Listato 3: Merge

```
1  #include <stdio.h>
2  #define MAX_LEN 50
3  int main( ) {
4    int V1[MAX_LEN], V2[MAX_LEN];
5    int V3[2*MAX_LEN];
6    int n, m;
7    int i, j, k;
8    /* acquisizione dei due vettori */
9    do {
10     printf("\nInserire il # di elementi del vettore V1
           (max %d), n = ", MAX_LEN);
11     scanf("%d",&n);
12   } while ((n <= 0) || (n > MAX_LEN));
13   printf("\nInserire gli elementi ordinati del vettore
       V1:\n");
14   for (i = 0; i < n; ++i) {
15     scanf("%d", &V1[i]);
16   }
17   do {
18     printf("\nInserire il # di elementi del vettore V2
           (max %d), m = ", MAX_LEN);
19     scanf("%d",&m);
20   } while ((m <= 0) || (m > MAX_LEN));
21   printf("\nInserire gli elementi ordinati del vettore
       V2:\n");
22   for (j = 0; j < m; ++j)
23     scanf("%d", &V2[j]);
24   }
25   /* fusione delle due sequenze */
26   i = 0;
27   j = 0;
28   k = 0;
29   while ((i<n) && (j<m)) {
30     if (V1[i] < V2[j]) {
31       V3[k] = V1[i];
32       i++;
33     } else {
34       V3[k] = V2[j];
```

```
35        j++;
36      }
37      k++;
38    }
39    while (i < n) {
40      V3[k] = V1[i];
41      i++;
42      k++;
43    }
44    while (j < m) {
45      V3[k] = V2[j];
46      j++;
47      k++;
48    }
49    printf("\nIl risultato della fusione dei due vettori
          e':\n");
50    for (k = 0; k < n + m; ++k) {
51      printf("%d ", V3[k]);
52    }
53    return 0;
54  }
```

3 Valutazione di un polinomio in un punto

Leggere un polinomio di grado n a coefficienti reali e successivamente valutarlo in un dato punto x.

3.1 Approccio alla soluzione

Nel caso in cui il punto in cui calcolare questo polinomio fosse noto a priori si poteva calcolare il risultato parziale mentre l'utente inseriva i valori senza doverli memorizzare in un vettore.

3.2 Soluzione 1: calcolo per monomi

Il primo approccio che sicuramente si può individuare è quello di valutare monomio per monomio l'espressione. In altre parole, sostituire ad x un certo valore, valutare le diverse potenze di x e moltiplicare per il corrispettivo coefficiente accumulando man mano il risultato in una variabile.

Listato 4: Valutazione polinomio: calcolo per monomi

```
1  #include <stdio.h>
2  #define MAX_LEN 100
3  int main() {
4    float V[MAX_LEN];
5    float pot, f, x;
6    int i, n;
7    do {
8      printf("\nInserire il grado del polinomio (max %d)
           , n = ", MAX_LEN - 1);
9      scanf("%d", &n);
10   } while ((n <= 0) || (n >= MAX_LEN));
11   printf("\nInserire i coefficienti del polinomio dal
       grado %d al termine noto: ", n);
12   for (i = 0; i <= n; ++i) {
13     scanf("%f", &V[i]);
14   }
15   printf("\nInserire il valore in cui valutare il
       polinomio, x = ");
16   scanf("%f", &x);
17   f = 0;
18   for (i = 0; i <= n; ++i) {
19     pot = 1;
20     /* calcola pot=x^i */
21     for (j = 0; j < i; ++j) {
22       pot = pot * x;
23     }
24     f = f + V[i] * pot;
25   }
26   printf("\nf(%f) = %f", x, f);
27   return 0;
28 }
```

3.3 Soluzione 2: costruzione progressiva della potenza

Iniziando a leggere il polinomio dal termine noto (il quale, considerato che il vettore avrà n+1 elementi, occuperà la posizione n-esima), si ha:

$$\begin{aligned} f(x) &= a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0 \\ &= \left(a_n \cdot x^n + \left(a_{n-1} \cdot x^{n-1} + \ldots + (a_1 \cdot x + (a_0))\right)\right) \\ &= \left(\left(\left((a_0) + a_1 \cdot x\right) + \ldots + a_{n-1} \cdot x^{n-1}\right) + a_n \cdot x^n\right) \end{aligned}$$

È quindi possibile costruire la potenza x^n mentre si sta iterando sul vettore senza doverla ricalcolare da x^0 ogni volta.

Listato 5: Valutazione polinomio: costruzione progressiva della potenza

```
1  #include <stdio.h>
2  #define MAX_LEN 100
3  int main() {
4    float V[MAX_LEN];
5    float pot, f, x;
6    int i, n;
7    do {
8      printf("\nInserire il grado del polinomio (max %d)
           , n = ", MAX_LEN - 1);
9      scanf("%d", &n);
10   } while ((n <= 0) || (n >= MAX_LEN));
11   printf("\nInserire i coefficienti del polinomio dal
       grado %d al termine noto: ", n);
12   for (i = 0; i <= n; ++i) {
13     scanf("%f", &V[i]);
14   }
15   printf("\nInserire il valore in cui valutare il
       polinomio, x = ");
16   scanf("%f", &x);
17   f = 0;
18   pot = 1;
19   for (i = n; i >= 0; i--) {
20     f = f + V[i] * pot;
21     pot = pot * x;
22   }
23   printf("\nf(%f) = %f", x, f);
24   return 0;
25 }
```

3.4 Soluzione 3: potenza implicita

Sempre sfruttando proprietà matematiche dei polinomi, possiamo riscrivere il polinomio come

$$\begin{aligned} f(x) &= a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + \ldots + a_1 \cdot x + a_0 \\ &= (((0 \cdot x + a_n) \cdot x + a_{n-1}) \cdot x + \ldots + a_1) \cdot x + a_0 \end{aligned}$$

In questo modo non è necessario calcolare la potenza x^n ma implicitamente viene calcolata moltiplicando il risultato parziale n volte per x.

Listato 6: Valutazione polinomio: potenza implicita

```
1  #include <stdio.h>
2  #define MAX_LEN 100
3  int main() {
4    float V[MAX_LEN];
5    float pot, f, x;
6    int i, n;
7    do {
8      printf("\nInserire il grado del polinomio (max %d)
           , n = ", MAX_LEN - 1);
9      scanf("%d", &n);
10   } while ((n <= 0) || (n >= MAX_LEN));
11   printf("\nInserire i coefficienti del polinomio dal
       grado %d al termine noto: ", n);
12   for (i = 0; i <= n; ++i) {
13     scanf("%f", &V[i]);
14   }
15   printf("\nInserire il valore in cui valutare il
       polinomio, x = ");
16   scanf("%f", &x);
17   f = 0;
18   for (i = 0; i <= n; ++i) {
19     f = f * x + V[i];
20   }
21   printf("\nf(%f) = %f", x, f);
22   return 0;
23 }
```

4 Ordinamento di vettori

Data una sequenza vettoriale V[n] permutare i suoi elementi in modo che risulti ordinata in senso non decrescente.
Esempio: $v[0] \leq v[1] \leq v[2] \leq \ldots \leq v[n-1]$.

4.1 Approccio alla soluzione

Esistono diversi algoritmi di ordinamento di vettori. Di seguito ne viene presentato uno, tra i più semplici: l'ordinamento per selezione.

4.2 Soluzione C - Selection Sort

L'idea è quella di trovare il minimo fra gli elementi nella sequenza e scambiarlo con l'elemento al primo posto. Trovare il minimo fra gli elementi tra il secondo e l'ultimo posto, e scambiarlo con il secondo e così via. Il metodo è chiamato *per selezione* perché si basa sulla ripetizione della selezione dell'elemento minore, tra quelli rimasti da ordinare.

Listato 7: Ordinamento per selezione

```
1  #include <stdio.h>
2  #define MAX_LEN 100
3  int main ( ) {
4    int V[MAX_LEN];
5    int n, min;
6    int i, j, jmin;
7    do {
8      printf("\nInserire il # di elementi del vettore V
           (max %d), n = ", MAX_LEN);
9      scanf("%d", &n);
10   }while ((n <= 0) || (n >= MAX_LEN));
11   printf("\nInserire gli elementi del vettore V:\n");
12   for (i = 0; i < n; ++i) {
13     scanf("%d", &V[i]);
14   }
15   for (i = 0; i < n; i++) {
16     jmin = i;
17     min = V[jmin];
18     for (j = i + 1; j < n; j++) {
19       if (V[j] < min) {
20         jmin = j;
21         min = V[j];
22       }
23     }
24   /* min ora contiene il minimo parziale, cioè il
        minimo calcolato dalla posizione i alla fine del
        vettore */
25     V[jmin] = V[i];
26     V[i] = min; /* scambio V[i] e V[jmin] */
27   }
28   printf("\n il vettore ordinato e':\n ");
29   for (i = 0; i < n; i++) {
30     printf("%d ",V[i]);
31   }
32   return 0;
33 }
```

4.3 Soluzione C - Insertion sort

L'idea di questo algoritmo non è molto diversa dal metodo con cui molti giocatori di poker ordinano le carte ricevute nella prima fase del gioco. Il vettore, di lunghezza n, lo si immagina logicamente partizionato in due parti:

- la prima parte già ordinata
- la seconda parte (contigua alla precedente) con elementi ancora da posizionare correttamente

Questo algoritmo prevede di allargare ad ogni iterazione la parte ordinata inserendo nella posizione corretta un elemento preso dalla parte non ordinata.

Listato 8: Ordinamento per inserzione

```
1  #include <stdio.h>
2  #define MAX_LEN 100
3  int V[MAX_LEN]; /* vettore da ordinare - var. globale
       */
4  int main( ) {
5    int i, j, n, aux;
6    do {
7      printf("\nInserire il # di elementi del vettore (
           max %d) V, n = ", MAX_LEN);
8      scanf("%d", &n);
9    } while ((n <= 0) || (n >= MAX_LEN));
10   printf("\nInserire gli elementi del vettore V:\n");
11   for (i = 0; i < n; ++i){
12     scanf("%d",&V[i]);
13   }
14   /* dal secondo al penultimo elemento */
15   for (i = 1; i < n; i++) {
16     aux = V[i]; /* elemento da inserire ordinato */
17     /* Shift verso dx della parte ordinata */
18     for (j = i - 1; j >= 0 && V[j] > aux; j--)
19       V[j+1] = V[j];
20     /* inserimento in ordine */
21     V[j+1] = aux;
22   }
23   printf("\nIl vettore ordinato e':\n ");
24   for (i = 0; i < n; i++)
25     printf("%d ",V[i]);
26   return 0;
27 }
```

4.4 Soluzione C - Bubblesort

L'idea di questo algoritmo è quella di confrontare i primi due elementi e se non sono ordinati li si scambia; poi si confrontano il secondo e il terzo e se non sono ordinati li si scambia e così via sino a confrontare penultimo e ultimo elemento.

Dopo aver scandito una prima volta tutto il vettore si è sicuri che l'elemento maggiore è nella cella più a destra, quindi si comincia un nuovo ciclo che confronta ancora a due a due le celle dalla prima all'ultima.

Se n è il numero di elementi del vettore, si itera questo processo di scansione per n-1 volte al termine delle quali il vettore risulterà completamente ordinato.

```
1  #include <stdio.h>
2  #define MAX_LEN 100
3  int V[MAX_LEN];
4  int main( ){
5    int i, j, n, aux;
6    do {
7      printf("\nInserire il # di elementi del vettore V
           (max %d), n = ", MAX_LEN);
8      scanf("%d", &n);
9    } while ((n <= 0) || (n > MAX_LEN));
10   printf("\nInserire gli elementi del vettore V:\n");
11   for (i = 0; i < n; ++i) {
12     scanf("%d", &V[i]);
13   }
14   /* L'ultimo controllo deve essere tra penultimo e
        ultimo. Quindi stop a n-1 */
15   for (i = 0; i < n - 1; i++) {
16   /* Dopo la i-esima iterazione, le ultime i posizioni
         sono già ordinate: posso quindi evitare di fare
        controlli su di esse */
17     for (j = 0; j < n - 1 - i; j++) {
18       if (V[j] > V[j + 1]) {
19         aux = V[j + 1];
20         V[j + 1] = V[j];
21         V[j] = aux;
22       }
23     }
24   }
25   printf("\nIl vettore ordinato e':\n ");
26   for (i = 0; i < n; i++)
27     printf("%d ", V[i]);
28   return 0;
29 }
```

Licenza e crediti

Crediti

Quest'opera contiene elementi tratti da materiale di Gerardo Pelosi redatto per il corso di Fondamenti di Informatica per Ingegneria dell'Automazione a.a. 2013/14.

Licenza beerware[1]

Quest'opera è stata redatta da Stefano Cherubin. Mantenendo questa nota, puoi fare quello che vuoi con quest'opera. Se ci dovessimo incontrare e tu ritenessi che quest'opera lo valga, in cambio puoi offrirmi una birra.

[1] http://people.freebsd.org/~phk/