



SAPIENZA
UNIVERSITÀ DI ROMA

Algoritmi

Esercizi 2 — Linguaggi di programmazione

Esercizi svolti sui capitoli 6–8: programmi in C++, Python e Java, dallo pseudo-codice al codice. Ogni esercizio ha la soluzione completa.

Fabio Furini

fabiofurini.github.io

Dipartimento di Ingegneria informatica, automatica e gestionale — Sapienza Università di Roma

Versione online e materiale scaricabile:

fabiofurini.github.io/algoritmi/
github.com/fabiofurini/algoritmi

Anno accademico 2026–2027

Indice

6	Introduzione a C++	2
7	Introduzione a Python	26
8	Introduzione a Java	40



Introduzione a C++

Negli esercizi di questo capitolo i programmi sono scritti, compilati ed eseguiti con le convenzioni del capitolo «Introduzione a C++», riassunte qui sotto.

Attenzione

- Il programma di un esercizio è salvato in un file `<name>.cpp`, compilato ed eseguito con i comandi

```
g++ -Wall -std=c++17 -o <name> <name>.cpp
./<name>
```

- Le istanze di test sono array di dimensione fissa dichiarati in `main`, ad esempio `int arr1[] = {9, 12, 3, 16, 19, 11, 12};`, e vengono passate alle funzioni tramite il puntatore al loro primo elemento (`int *a`) insieme alla loro dimensione `n`.
- Gli elementi di un array di dimensione n sono `a[0]`, `a[1]`, $\dots$, `a[n-1]`: la componente `a[i]` di un'istanza scritta come `a = [...]` nel testo corrisponde ad `a[i-1]` nel codice.
- I risultati che una funzione deve restituire oltre al valore di ritorno sono passati per riferimento (`int& x`, `double& x`).
- Quando si stampa un array, ogni valore è seguito da uno spazio, quindi le righe stampate terminano con uno spazio.

Esercizio 6.1 — Introduzione a C++ (E1)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di trovare il valore massimo nell'array.

Domande

1. Scrivere un programma C++ completo che trova il valore massimo in un array a di n numeri interi. Il programma deve:
 - Definire una funzione `int ArrayMaxValue(int n, int *a)` che implementa l'algoritmo

- Nella funzione `main()`, creare gli array di test e chiamare la funzione
- Stampare i risultati per ogni caso di test

2. Eseguire il programma sulle seguenti istanze:

- Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
- Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
- Istanza 3: $a = [42]$ con $n = 1$

Mostrare i comandi di compilazione completi e l'output per tutti i casi di test.

3. Modificare il programma creando una nuova funzione `int ArrayMaxValueWithIndex(int n, int *a, int& idx)` che restituisce il valore massimo e memorizza il suo indice nel parametro per riferimento `idx`. Eseguire questa versione modificata e mostrare l'output.

Soluzione

1. Complete C++ program:

```

1  #include <iostream>
2  using namespace std;
3
4  // Funzione che trova il valore massimo di un array
5  int ArrayMaxValue(int n, int *a)
6  {
7      int max = a[0]; // Inizializza con il primo elemento
8
9      for (int i = 1; i < n; i++)
10     {
11         if (a[i] > max)
12         {
13             max = a[i]; // Aggiorna il massimo
14         }
15     }
16
17     return max;
18 }
19
20 int main()
21 {
22     // Caso di prova 1
23     int arr1[] = {9, 12, 3, 16, 19, 11, 12};
24     int n1 = 7;
25
26     cout << "Prova 1 - Array: ";
27     for (int i = 0; i < n1; i++)
28     {
29         cout << arr1[i] << " ";
30     }

```

```
31     cout << endl;
32
33     int max1 = ArrayMaxValue(n1, arr1);
34     cout << "Valore massimo: " << max1 << endl;
35     cout << endl;
36
37     // Caso di prova 2
38     int arr2[] = {5, 2, 8, 1};
39     int n2 = 4;
40
41     cout << "Prova 2 - Array: ";
42     for (int i = 0; i < n2; i++)
43     {
44         cout << arr2[i] << " ";
45     }
46     cout << endl;
47
48     int max2 = ArrayMaxValue(n2, arr2);
49     cout << "Valore massimo: " << max2 << endl;
50     cout << endl;
51
52     // Caso di prova 3
53     int arr3[] = {42};
54     int n3 = 1;
55
56     cout << "Prova 3 - Array: ";
57     for (int i = 0; i < n3; i++)
58     {
59         cout << arr3[i] << " ";
60     }
61     cout << endl;
62
63     int max3 = ArrayMaxValue(n3, arr3);
64     cout << "Valore massimo: " << max3 << endl;
65
66     return 0;
67 }
```

2. Compilazione ed esecuzione:

Per compilare il programma:

```
g++ -Wall -std=c++17 -o max_value max_value.cpp
```

Per eseguire il programma:

```
./max_value
```

Output:

Prova 1 - Array: 9 12 3 16 19 11 12
Valore massimo: 19

Prova 2 - Array: 5 2 8 1
Valore massimo: 8

Prova 3 - Array: 42
Valore massimo: 42

3. Programma modificato con l'indice:

```
1 #include <iostream>
2 using namespace std;
3
4 // Funzione che trova il valore massimo e il suo indice
5 int ArrayMaxValueWithIndex(int n, int *a, int& idx)
6 {
7     int max = a[0];
8     idx = 0;
9
10    for (int i = 1; i < n; i++)
11    {
12        if (a[i] > max)
13        {
14            max = a[i];
15            idx = i; // Memorizza l'indice
16        }
17    }
18
19    return max;
20 }
21
22 int main()
23 {
24     int arr[] = {9, 12, 3, 16, 19, 11, 12};
25     int n = 7;
26     int index;
27
28     cout << "Array: ";
29     for (int i = 0; i < n; i++)
30     {
31         cout << arr[i] << " ";
32     }
33     cout << endl;
34
35     int max = ArrayMaxValueWithIndex(n, arr, index);
36
37     cout << "Valore massimo: " << max << endl;
38     cout << "Posizione (indice): " << index << endl;
39
40     return 0;
41 }
```

Output:

Array: 9 12 3 16 19 11 12
 Valore massimo: 19
 Posizione (indice): 4

Nota: il valore massimo 19 si trova all'indice 4 (l'elemento di posizione 5, poiché in C++ gli array sono indicizzati a partire da 0). Se il valore massimo compare più di una volta, la funzione memorizza l'indice della sua prima occorrenza, perché `max` e `idx` vengono aggiornati solo quando `a[i] > max` (disuguaglianza stretta).

Esercizio 6.2 — Introduzione a C++ (E2)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di trovare il suo valore minimo.

Domande

1. Scrivere un programma C++ completo che trova il valore minimo in un array a di n numeri interi. Il programma deve:
 - Definire una funzione `int ArrayMinValue(int n, int *a)` che implementa l'algoritmo
 - Nella funzione `main()`, creare gli array di test e chiamare la funzione
 - Stampare i risultati per ogni caso di test
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [15, 7, 22, 4]$ con $n = 4$
 - Istanza 3: $a = [100]$ con $n = 1$

Mostrare i comandi di compilazione completi e l'output per tutti i casi di test.

3. Modificare il programma creando una nuova funzione `void ArrayMaxMin(int n, int *a, int& max, int& min)` che trova sia il valore massimo sia il valore minimo con un solo passaggio sull'array. Eseguire questa versione modificata e mostrare l'output.

Soluzione

1. **Complete C++ program:**

```
1 #include <iostream>
2 using namespace std;
3
4 // Funzione che trova il valore minimo di un array
5 int ArrayMinValue(int n, int *a)
6 {
7     int m = a[0]; // Inizializza con il primo elemento
8
9     for (int i = 1; i < n; i++)
10     {
11         if (a[i] < m)
12         {
13             m = a[i]; // Aggiorna il minimo
14         }
15     }
16
17     return m;
18 }
19
20 int main()
21 {
22     // Caso di prova 1
23     int arr1[] = {9, 12, 3, 16, 19, 11, 12};
24     int n1 = 7;
25
26     cout << "Prova 1 - Array: ";
27     for (int i = 0; i < n1; i++)
28     {
29         cout << arr1[i] << " ";
30     }
31     cout << endl;
32
33     int min1 = ArrayMinValue(n1, arr1);
34     cout << "Valore minimo: " << min1 << endl;
35     cout << endl;
36
37     // Caso di prova 2
38     int arr2[] = {15, 7, 22, 4};
39     int n2 = 4;
40
41     cout << "Prova 2 - Array: ";
42     for (int i = 0; i < n2; i++)
43     {
44         cout << arr2[i] << " ";
45     }
46     cout << endl;
47
48     int min2 = ArrayMinValue(n2, arr2);
49     cout << "Valore minimo: " << min2 << endl;
50     cout << endl;
51
52     // Caso di prova 3
53     int arr3[] = {100};
54     int n3 = 1;
55
56     cout << "Prova 3 - Array: ";
```

```
57     for (int i = 0; i < n3; i++)
58     {
59         cout << arr3[i] << " ";
60     }
61     cout << endl;
62
63     int min3 = ArrayMinValue(n3, arr3);
64     cout << "Valore minimo: " << min3 << endl;
65
66     return 0;
67 }
```

2. Compilazione ed esecuzione:

Per compilare il programma:

```
g++ -Wall -std=c++17 -o min_value min_value.cpp
```

Per eseguire il programma:

```
./min_value
```

Output:

Prova 1 - Array: 9 12 3 16 19 11 12

Valore minimo: 3

Prova 2 - Array: 15 7 22 4

Valore minimo: 4

Prova 3 - Array: 100

Valore minimo: 100

3. Programma modificato per trovare sia il massimo sia il minimo:

```
1  #include <iostream>
2  using namespace std;
3
4  // Funzione che trova sia il massimo sia il minimo
5  void ArrayMaxMin(int n, int *a, int& max, int& min)
6  {
7      max = a[0];
8      min = a[0];
9
10     for (int i = 1; i < n; i++)
11     {
12         if (a[i] > max)
13         {
14             max = a[i];
```

```

15     }
16     if (a[i] < min)
17     {
18         min = a[i];
19     }
20 }
21 }
22
23 int main()
24 {
25     int arr[] = {9, 12, 3, 16, 19, 11, 12};
26     int n = 7;
27     int maximum, minimum;
28
29     cout << "Array: ";
30     for (int i = 0; i < n; i++)
31     {
32         cout << arr[i] << " ";
33     }
34     cout << endl;
35
36     ArrayMaxMin(n, arr, maximum, minimum);
37
38     cout << "Valore massimo: " << maximum << endl;
39     cout << "Valore minimo: " << minimum << endl;
40
41     return 0;
42 }

```

Output:

```

Array: 9 12 3 16 19 11 12
Valore massimo: 19
Valore minimo: 3

```

Nota: questa implementazione trova sia il valore massimo sia il valore minimo con un solo passaggio sull'array. Il numero di confronti è lo stesso che con due funzioni separate, $2(n - 1)$, ma l'array viene scorso una volta sola invece di due.

Esercizio 6.3 — Introduzione a C++ (E3)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di calcolare la somma e la media di tutti i suoi elementi.

Domande

1. Scrivere un programma C++ completo che calcola la somma e la media degli elementi di un array a di n numeri interi. Il programma deve:

- Definire una funzione `int ArraySum(int n, int *a)` che calcola la somma degli elementi dell'array
- Nella funzione `main()`, creare gli array di test e chiamare la funzione
- Calcolare e stampare sia la somma sia la media per ogni caso di test

2. Eseguire il programma sulle seguenti istanze:

- Istanza 1: $a = [5, 12, 3, 8]$ con $n = 4$
- Istanza 2: $a = [10, 20, 30]$ con $n = 3$
- Istanza 3: $a = [7]$ con $n = 1$

Mostrare i comandi di compilazione completi e l'output per tutti i casi di test.

3. Modificare il programma creando una nuova funzione `void ArraySumAndAverage(int n, int *a, int& sum, double& avg)` che calcola sia la somma sia la media in un'unica funzione. Eseguire questa versione modificata e mostrare l'output.

Soluzione

1. Complete C++ program:

```

1  #include <iostream>
2  #include <iomanip>
3  using namespace std;
4
5  // Funzione che calcola la somma degli elementi dell'array
6  int ArraySum(int n, int *a)
7  {
8      int S = 0;
9
10     for (int i = 0; i < n; i++)
11     {
12         S = S + a[i];
13     }
14
15     return S;
16 }
17
18 int main()
19 {
20     // Caso di prova 1
21     int arr1[] = {5, 12, 3, 8};
22     int n1 = 4;
23
24     cout << "Prova 1 - Array: ";
25     for (int i = 0; i < n1; i++)
26     {
27         cout << arr1[i] << " ";
28     }
29     cout << endl;

```

```

30
31     int sum1 = ArraySum(n1, arr1);
32     double avg1 = (double)sum1 / n1;
33     cout << "Somma: " << sum1 << endl;
34     cout << "Media: " << fixed << setprecision(2)
35         << avg1 << endl;
36     cout << endl;
37
38     // Caso di prova 2
39     int arr2[] = {10, 20, 30};
40     int n2 = 3;
41
42     cout << "Prova 2 - Array: ";
43     for (int i = 0; i < n2; i++)
44     {
45         cout << arr2[i] << " ";
46     }
47     cout << endl;
48
49     int sum2 = ArraySum(n2, arr2);
50     double avg2 = (double)sum2 / n2;
51     cout << "Somma: " << sum2 << endl;
52     cout << "Media: " << fixed << setprecision(2)
53         << avg2 << endl;
54     cout << endl;
55
56     // Caso di prova 3
57     int arr3[] = {7};
58     int n3 = 1;
59
60     cout << "Prova 3 - Array: ";
61     for (int i = 0; i < n3; i++)
62     {
63         cout << arr3[i] << " ";
64     }
65     cout << endl;
66
67     int sum3 = ArraySum(n3, arr3);
68     double avg3 = (double)sum3 / n3;
69     cout << "Somma: " << sum3 << endl;
70     cout << "Media: " << fixed << setprecision(2)
71         << avg3 << endl;
72
73     return 0;
74 }

```

2. Compilazione ed esecuzione:

Per compilare il programma:

```
g++ -Wall -std=c++17 -o array_sum array_sum.cpp
```

Per eseguire il programma:

```
./array_sum
```

Output:

Prova 1 - Array: 5 12 3 8

Somma: 28

Media: 7.00

Prova 2 - Array: 10 20 30

Somma: 60

Media: 20.00

Prova 3 - Array: 7

Somma: 7

Media: 7.00

3. Programma modificato con la funzione combinata:

```
1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 // Funzione che calcola sia la somma sia la media
6 void ArraySumAndAverage(int n, int *a, int& sum, double& avg)
7 {
8     sum = 0;
9
10    for (int i = 0; i < n; i++)
11    {
12        sum = sum + a[i];
13    }
14
15    avg = (double)sum / n;
16 }
17
18 int main()
19 {
20     int arr[] = {5, 12, 3, 8};
21     int n = 4;
22     int sum;
23     double average;
24
25     cout << "Array: ";
26     for (int i = 0; i < n; i++)
27     {
28         cout << arr[i] << " ";
29     }
30     cout << endl;
31
32     ArraySumAndAverage(n, arr, sum, average);
33 }
```

```

34     cout << "Somma: " << sum << endl;
35     cout << "Media: " << fixed << setprecision(2)
36         << average << endl;
37
38     return 0;
39 }

```

Output:

Array: 5 12 3 8

Somma: 28

Media: 7.00

Nota: il manipolatore `setprecision(2)` serve a formattare la media con 2 cifre decimali. La conversione (`double`) garantisce la divisione in virgola mobile: senza di essa, `sum / n` sarebbe la divisione intera. Poiché la media divide per n , le funzioni richiedono $n \geq 1$.

Esercizio 6.4 — Introduzione a C++ (E4)

Dato un array a di n numeri interi e un valore m , consideriamo il problema di determinare se il valore m è presente o meno nell'array.

Domande

1. Scrivere un programma C++ completo che implementa l'algoritmo di ricerca lineare per un array a di n numeri interi. Il programma deve:
 - Definire una funzione `int LinearSearch(int n, int *a, int m)` che cerca il valore m nell'array a
 - Restituire l'indice dell'elemento se viene trovato, oppure `-1` se non viene trovato
 - Nella funzione `main()`, creare un array di test e cercare valori diversi
 - Stampare i risultati di ogni ricerca
2. Eseguire il programma con la seguente istanza:
 - Array: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Ricerca di: $m = 16$ (elemento presente)
 - Ricerca di: $m = 25$ (elemento non presente)
 - Ricerca di: $m = 9$ (primo elemento)
 - Ricerca di: $m = 12$ (elemento duplicato)

Mostrare i comandi di compilazione completi e l'output per tutti i casi di test.

3. Modificare il programma creando una funzione `int CountOccurrences(int n, int *a, int m)` che conta quante volte il valore `m` compare nell'array. Eseguire questa versione modificata con il valore $m = 12$ nell'array di test e mostrare l'output.

Soluzione

1. Complete C++ program:

```

1  #include <iostream>
2  using namespace std;
3
4  // Funzione che esegue la ricerca lineare
5  int LinearSearch(int n, int *a, int m)
6  {
7      for (int i = 0; i < n; i++)
8      {
9          if (a[i] == m)
10         {
11             return i; // Trovato: restituisce l'indice
12         }
13     }
14
15     return -1; // Non trovato
16 }
17
18 int main()
19 {
20     int arr[] = {9, 12, 3, 16, 19, 11, 12};
21     int n = 7;
22
23     cout << "Array: ";
24     for (int i = 0; i < n; i++)
25     {
26         cout << arr[i] << " ";
27     }
28     cout << endl << endl;
29
30     // Prova 1: ricerca di 16 (presente)
31     int m1 = 16;
32     int result1 = LinearSearch(n, arr, m1);
33     cout << "Ricerca di " << m1 << ": ";
34     if (result1 != -1)
35     {
36         cout << "Trovato all'indice " << result1 << endl;
37     }
38     else
39     {
40         cout << "Non trovato" << endl;
41     }
42
43     // Prova 2: ricerca di 25 (assente)
44     int m2 = 25;

```

```
45     int result2 = LinearSearch(n, arr, m2);
46     cout << "Ricerca di " << m2 << ": ";
47     if (result2 != -1)
48     {
49         cout << "Trovato all'indice " << result2 << endl;
50     }
51     else
52     {
53         cout << "Non trovato" << endl;
54     }
55
56     // Prova 3: ricerca di 9 (primo elemento)
57     int m3 = 9;
58     int result3 = LinearSearch(n, arr, m3);
59     cout << "Ricerca di " << m3 << ": ";
60     if (result3 != -1)
61     {
62         cout << "Trovato all'indice " << result3 << endl;
63     }
64     else
65     {
66         cout << "Non trovato" << endl;
67     }
68
69     // Prova 4: ricerca di 12 (ripetuto)
70     int m4 = 12;
71     int result4 = LinearSearch(n, arr, m4);
72     cout << "Ricerca di " << m4 << ": ";
73     if (result4 != -1)
74     {
75         cout << "Trovato all'indice " << result4
76             << " (prima occorrenza)" << endl;
77     }
78     else
79     {
80         cout << "Non trovato" << endl;
81     }
82
83     return 0;
84 }
```

2. Compilazione ed esecuzione:

Per compilare il programma:

```
g++ -Wall -std=c++17 -o linear_search linear_search.cpp
```

Per eseguire il programma:

```
./linear_search
```

Output:

Array: 9 12 3 16 19 11 12

Ricerca di 16: Trovato all'indice 3

Ricerca di 25: Non trovato

Ricerca di 9: Trovato all'indice 0

Ricerca di 12: Trovato all'indice 1 (prima occorrenza)

3. Programma modificato per contare le occorrenze:

```

1  #include <iostream>
2  using namespace std;
3
4  // Funzione che conta le occorrenze del valore m
5  int CountOccurrences(int n, int *a, int m)
6  {
7      int count = 0;
8
9      for (int i = 0; i < n; i++)
10     {
11         if (a[i] == m)
12         {
13             count++;
14         }
15     }
16
17     return count;
18 }
19
20 int main()
21 {
22     int arr[] = {9, 12, 3, 16, 19, 11, 12};
23     int n = 7;
24
25     cout << "Array: ";
26     for (int i = 0; i < n; i++)
27     {
28         cout << arr[i] << " ";
29     }
30     cout << endl;
31
32     int x = 12;
33     int count = CountOccurrences(n, arr, x);
34
35     cout << "Il valore " << x << " compare " << count
36         << " volte nell'array" << endl;
37
38     return 0;
39 }
```

Output:

Array: 9 12 3 16 19 11 12

Il valore 12 compare 2 volte nell'array

Nota: la funzione `LinearSearch` restituisce l'indice della prima occorrenza quando ci sono duplicati. La funzione `CountOccurrences` scorre l'intero array per contare tutte le occorrenze.

Esercizio 6.5 — Introduzione a C++ (E5)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di calcolare con un solo passaggio il massimo, il minimo, la somma e la media di tutti gli elementi dell'array.

Domande

1. Scrivere un programma C++ completo che calcola tutte le statistiche di un array a di n numeri interi con un solo passaggio. Il programma deve:
 - Definire una funzione `void ArrayStatistics(int n, int *a, int& max, int& min, int& sum, double& avg)` che calcola tutte e quattro le statistiche
 - Nella funzione `main()`, creare gli array di test e chiamare la funzione
 - Stampare tutte le statistiche per ogni caso di test
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$

Mostrare i comandi di compilazione completi e l'output per tutti i casi di test.

3. Discutere i vantaggi di calcolare tutte le statistiche con un solo passaggio rispetto a calcolarle separatamente con più funzioni.

Soluzione

1. Complete C++ program:

```

1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4
5 // Funzione che calcola tutte le statistiche in una sola passata
6 void ArrayStatistics(int n, int *a, int& max, int& min,
7                     int& sum, double& avg)
8 {
9     // Inizializza con il primo elemento
10    max = a[0];
11    min = a[0];
12    sum = a[0];

```

```
13
14 // Elabora gli elementi rimanenti
15 for (int i = 1; i < n; i++)
16 {
17     if (a[i] > max)
18     {
19         max = a[i];
20     }
21     if (a[i] < min)
22     {
23         min = a[i];
24     }
25     sum = sum + a[i];
26 }
27
28 // Calcola la media
29 avg = (double)sum / n;
30 }
31
32 int main()
33 {
34     // Caso di prova 1
35     int arr1[] = {9, 12, 3, 16, 19, 11, 12};
36     int n1 = 7;
37     int max1, min1, sum1;
38     double avg1;
39
40     cout << "Prova 1 - Array: ";
41     for (int i = 0; i < n1; i++)
42     {
43         cout << arr1[i] << " ";
44     }
45     cout << endl;
46
47     ArrayStatistics(n1, arr1, max1, min1, sum1, avg1);
48
49     cout << "Massimo: " << max1 << endl;
50     cout << "Minimo: " << min1 << endl;
51     cout << "Somma: " << sum1 << endl;
52     cout << "Media: " << fixed << setprecision(2)
53         << avg1 << endl;
54     cout << endl;
55
56     // Caso di prova 2
57     int arr2[] = {5, 2, 8, 1};
58     int n2 = 4;
59     int max2, min2, sum2;
60     double avg2;
61
62     cout << "Prova 2 - Array: ";
63     for (int i = 0; i < n2; i++)
64     {
65         cout << arr2[i] << " ";
66     }
67     cout << endl;
68 }
```

```
69     ArrayStatistics(n2, arr2, max2, min2, sum2, avg2);
70
71     cout << "Massimo: " << max2 << endl;
72     cout << "Minimo: " << min2 << endl;
73     cout << "Somma: " << sum2 << endl;
74     cout << "Media: " << fixed << setprecision(2)
75           << avg2 << endl;
76
77     return 0;
78 }
```

2. Compilazione ed esecuzione:

Per compilare il programma:

```
g++ -Wall -std=c++17 -o array_stats array_stats.cpp
```

Per eseguire il programma:

```
./array_stats
```

Output:

```
Prova 1 - Array: 9 12 3 16 19 11 12
Massimo: 19
Minimo: 3
Somma: 82
Media: 11.71
```

```
Prova 2 - Array: 5 2 8 1
Massimo: 8
Minimo: 1
Somma: 16
Media: 4.00
```

3. Vantaggi dell'approccio a un solo passaggio:

Vantaggi:

- **Meno scorrimenti dell'array:** l'array viene scorso una sola volta invece di una volta per ogni statistica
- **Codice più compatto:** una sola funzione e un solo ciclo invece di più chiamate di funzione
- **Migliore per dati in flusso:** può elaborare dati che arrivano in sequenza o che non stanno interamente in memoria

- **Minore sovraccarico:** una sola inizializzazione e terminazione di ciclo invece di più cicli

Svantaggi:

- **Meno modulare:** combina più operazioni in una sola funzione, che può essere più difficile da mantenere
- **Meno flessibile:** se serve una sola statistica, la funzione le calcola comunque tutte
- **Riutilizzabilità del codice:** non è facile riutilizzare i singoli calcoli delle statistiche in altri contesti

Raccomandazione: l'approccio a un solo passaggio è preferibile quando:

- servono tutte le statistiche insieme;
- si lavora con array grandi, per i quali è importante ridurre al minimo gli accessi all'array;
- si elaborano dati in flusso o dati che non stanno in memoria.

Le funzioni separate sono migliori quando:

- servono solo statistiche specifiche;
- la chiarezza e la modularità del codice hanno priorità maggiore;
- le singole funzioni possono essere riutilizzate in contesti diversi.

Esercizio 6.6 — Introduzione a C++ (E6)

Dato un array bidimensionale a di $m \times n$ numeri interi, consideriamo il problema di calcolare la somma di tutti i suoi elementi.

Domande

1. Scrivere un programma C++ completo che calcola la somma di tutti gli elementi di un array bidimensionale a di $m \times n$ interi. Il programma deve:
 - Definire una funzione `int SumMatrix(int m, int n, int **matrix)` che calcola la somma
 - Nella funzione `main()`, allocare le matrici di test con la memoria dinamica, inizializzarle, chiamare la funzione e stampare i risultati
 - Liberare tutta la memoria allocata dopo l'uso
2. Eseguire il programma sulle seguenti istanze:

- Istanza 1: una matrice 3×4 con valori sequenziali $a[i][j] = (i - 1) \cdot n + j$ per $i \in \{1, 2, 3\}$ e $j \in \{1, 2, 3, 4\}$ (righe e colonne numerate da 1, come nello pseudocodice), cioè

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{bmatrix}$$

- Istanza 2: una matrice 2×3 con valori $\begin{bmatrix} 3 & 1 & 4 \\ 1 & 5 & 9 \end{bmatrix}$

Mostrare i comandi di compilazione completi e l'output per entrambi i casi di test.

3. Modificare il programma aggiungendo una funzione `void RowSums(int m, int n, int **matrix, int *rowSums)` che calcola la somma di ogni riga e memorizza i risultati nell'array `rowSums`. Eseguire questa versione modificata sull'Istanza 1 e mostrare l'output.

Soluzione

1. Complete C++ program:

```

1 #include <iostream>
2 using namespace std;
3
4 // Funzione che calcola la somma di tutti gli elementi di una matrice
5 int SumMatrix(int m, int n, int **matrix)
6 {
7     int sum = 0;
8     for (int i = 0; i < m; i++)
9     {
10         for (int j = 0; j < n; j++)
11         {
12             sum += matrix[i][j];
13         }
14     }
15     return sum;
16 }
17
18 int main()
19 {
20     // Caso di prova 1: matrice 3x4 con valori consecutivi
21     int m1 = 3, n1 = 4;
22
23     int** mat1 = new int*[m1];
24     for (int i = 0; i < m1; i++)
25     {
26         mat1[i] = new int[n1];
27     }
28
29     int val = 1;
30     for (int i = 0; i < m1; i++)
31     {

```

```

32     for (int j = 0; j < n1; j++)
33     {
34         mat1[i][j] = val++;
35     }
36 }
37
38 cout << "Prova 1 - Matrice " << m1 << "x" << n1 << "." << endl;
39 for (int i = 0; i < m1; i++)
40 {
41     for (int j = 0; j < n1; j++)
42     {
43         cout << mat1[i][j] << " ";
44     }
45     cout << endl;
46 }
47 cout << "Somma: " << SumMatrix(m1, n1, mat1) << endl << endl;
48
49 for (int i = 0; i < m1; i++)
50 {
51     delete[] mat1[i];
52 }
53 delete[] mat1;
54
55 // Caso di prova 2: matrice 2x3
56 int m2 = 2, n2 = 3;
57 int data[2][3] = {{3, 1, 4}, {1, 5, 9}};
58
59 int** mat2 = new int*[m2];
60 for (int i = 0; i < m2; i++)
61 {
62     mat2[i] = new int[n2];
63     for (int j = 0; j < n2; j++)
64     {
65         mat2[i][j] = data[i][j];
66     }
67 }
68
69 cout << "Prova 2 - Matrice " << m2 << "x" << n2 << "." << endl;
70 for (int i = 0; i < m2; i++)
71 {
72     for (int j = 0; j < n2; j++)
73     {
74         cout << mat2[i][j] << " ";
75     }
76     cout << endl;
77 }
78 cout << "Somma: " << SumMatrix(m2, n2, mat2) << endl;
79
80 for (int i = 0; i < m2; i++)
81 {
82     delete[] mat2[i];
83 }
84 delete[] mat2;
85
86 return 0;
87 }

```

2. Compilazione ed esecuzione:

Per compilare il programma:

```
g++ -Wall -std=c++17 -o sum_matrix sum_matrix.cpp
```

Per eseguire il programma:

```
./sum_matrix
```

Output:

Prova 1 - Matrice 3x4:

```
1 2 3 4
5 6 7 8
9 10 11 12
Somma: 78
```

Prova 2 - Matrice 2x3:

```
3 1 4
1 5 9
Somma: 23
```

3. Funzione per le somme delle righe:

```
1 #include <iostream>
2 using namespace std;
3
4 // Funzione che calcola la somma di tutti gli elementi di una matrice
5 int SumMatrix(int m, int n, int **matrix)
6 {
7     int sum = 0;
8     for (int i = 0; i < m; i++)
9     {
10         for (int j = 0; j < n; j++)
11         {
12             sum += matrix[i][j];
13         }
14     }
15     return sum;
16 }
17
18 // Funzione che calcola la somma di ogni riga
19 void RowSums(int m, int n, int **matrix, int *rowSums)
20 {
21     for (int i = 0; i < m; i++)
22     {
```

```

23     rowSums[i] = 0;
24     for (int j = 0; j < n; j++)
25     {
26         rowSums[i] += matrix[i][j];
27     }
28 }
29 }
30
31 int main()
32 {
33     // Caso di prova 1: matrice 3x4 con valori consecutivi
34     int m1 = 3, n1 = 4;
35
36     int** mat1 = new int*[m1];
37     for (int i = 0; i < m1; i++)
38     {
39         mat1[i] = new int[n1];
40     }
41
42     int val = 1;
43     for (int i = 0; i < m1; i++)
44     {
45         for (int j = 0; j < n1; j++)
46         {
47             mat1[i][j] = val++;
48         }
49     }
50
51     cout << "Matrice " << m1 << "x" << n1 << ":" << endl;
52     for (int i = 0; i < m1; i++)
53     {
54         for (int j = 0; j < n1; j++)
55         {
56             cout << mat1[i][j] << " ";
57         }
58         cout << endl;
59     }
60     cout << "Somma totale: " << SumMatrix(m1, n1, mat1) << endl << endl;
61
62     int* rowSums = new int[m1];
63     RowSums(m1, n1, mat1, rowSums);
64
65     cout << "Somme delle righe:" << endl;
66     for (int i = 0; i < m1; i++)
67     {
68         cout << "Riga " << (i + 1) << ": " << rowSums[i] << endl;
69     }
70
71     delete[] rowSums;
72     for (int i = 0; i < m1; i++)
73     {
74         delete[] mat1[i];
75     }
76     delete[] mat1;
77
78     return 0;

```

```
79 }
```

Output:

Matrice 3x4:

1 2 3 4

5 6 7 8

9 10 11 12

Somma totale: 78

Somme delle righe:

Riga 1: 10

Riga 2: 26

Riga 3: 42

Introduzione a Python

Negli esercizi di questo capitolo i programmi sono scritti ed eseguiti con le convenzioni del capitolo «Introduzione a Python», riassunte qui sotto.

Attenzione

- Il programma di un esercizio è salvato in un file `<name>.py` ed eseguito con il comando `python3 <name>.py`
- Le istanze di test sono liste scritte nel codice, ad esempio `arr1 = [9, 12, 3, 16, 19, 11, 12]`, e sono passate alle funzioni senza la loro dimensione: una funzione ottiene la dimensione n di una lista `a` con `len(a)`.
- Gli elementi di una lista di dimensione n sono `a[0]`, `a[1]`, ..., `a[n-1]`: la componente `a[i]` di un'istanza scritta come `a = [...]` nel testo corrisponde ad `a[i-1]` nel codice, e gli indici stampati dai programmi partono da 0.
- Le istruzioni principali sono scritte nel blocco `if __name__ == "__main__":`; una lista si stampa con `print`, che la visualizza fra parentesi quadre.
- I risultati che una funzione deve restituire insieme sono restituiti come una tupla (`return x, y`) e ricevuti con un assegnamento multiplo (`x, y = f(a)`).
- Il formato `:.4f` dentro una f-string visualizza un numero in virgola mobile con esattamente 4 cifre dopo la virgola decimale.

Esercizio 7.1 — Introduzione a Python (E1)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di trovare il valore massimo nell'array.

Domande

1. Scrivere un programma Python completo che trova il valore massimo in una lista a di n numeri interi. Il programma deve:

- definire una funzione `array_max_value(a)` che implementa l'algoritmo;
- nel blocco `__main__`, creare le liste di test e chiamare la funzione;
- stampare i risultati per ogni caso di test.

2. Eseguire il programma sulle seguenti istanze:

- Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
- Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
- Istanza 3: $a = [42]$ con $n = 1$

Mostrare il comando che esegue il programma e l'output per tutti i casi di test.

3. Modificare la funzione in modo che restituisca sia il valore massimo sia il suo indice nella lista. Eseguire la versione modificata sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Python completo:

```

1 def array_max_value(a):
2     """Trova e restituisce il valore massimo della lista a."""
3     max_val = a[0] # Inizializza con il primo elemento
4
5     for i in range(1, len(a)):
6         if a[i] > max_val:
7             max_val = a[i] # Aggiorna il massimo
8
9     return max_val
10
11
12 if __name__ == "__main__":
13     # Caso di prova 1
14     arr1 = [9, 12, 3, 16, 19, 11, 12]
15     print(f"Prova 1 - Lista: {arr1}")
16     print(f"Valore massimo: {array_max_value(arr1)}")
17     print()
18
19     # Caso di prova 2
20     arr2 = [5, 2, 8, 1]
21     print(f"Prova 2 - Lista: {arr2}")
22     print(f"Valore massimo: {array_max_value(arr2)}")
23     print()
24
25     # Caso di prova 3
26     arr3 = [42]
27     print(f"Prova 3 - Lista: {arr3}")
28     print(f"Valore massimo: {array_max_value(arr3)}")

```

Per l'Istanza 3 il ciclo `for i in range(1, 1)` non esegue nessuna iterazione, e la funzione restituisce l'unico elemento `a[0]`.

2. **Esecuzione.** Se il programma è salvato nel file `max_value.py`, lo si esegue con il comando

```
python3 max_value.py
```

Output:

```
Prova 1 - Lista: [9, 12, 3, 16, 19, 11, 12]
Valore massimo: 19
```

```
Prova 2 - Lista: [5, 2, 8, 1]
Valore massimo: 8
```

```
Prova 3 - Lista: [42]
Valore massimo: 42
```

3. **Funzione modificata che restituisce il valore massimo e il suo indice:**

```
1 def array_max_value_with_index(a):
2     """Restituisce (valore_massimo, indice) dell'elemento massimo della
3     lista a."""
4     max_val = a[0]
5     idx = 0
6
7     for i in range(1, len(a)):
8         if a[i] > max_val:
9             max_val = a[i]
10            idx = i # Memorizza l'indice
11
12    return max_val, idx
13
14 if __name__ == "__main__":
15     arr1 = [9, 12, 3, 16, 19, 11, 12]
16     max_val, idx = array_max_value_with_index(arr1)
17     print(f"Valore massimo: {max_val}, all'indice: {idx}")
```

Output:

```
Valore massimo: 19, all'indice: 4
```

Per l'Istanza 1 la funzione restituisce la tupla (19, 4): il valore massimo 19 è all'indice 4 della lista (indici da 0), cioè nella posizione 5 dell'array `a[1..7]` del testo. Se il valore massimo compare più volte, la funzione restituisce l'indice della sua prima occorrenza, perché `max_val` e `idx` sono aggiornati solo quando `a[i] > max_val` (disuguaglianza stretta).

Esercizio 7.2 — Introduzione a Python (E2)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di trovare il valore minimo nell'array.

Domande

1. Scrivere un programma Python completo che trova il valore minimo in una lista a di n numeri interi. Il programma deve:
 - definire una funzione `array_min_value(a)` che implementa l'algoritmo;
 - nel blocco `__main__`, creare le liste di test e chiamare la funzione;
 - stampare i risultati per ogni caso di test.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare il comando che esegue il programma e l'output per tutti i casi di test.

3. Modificare la funzione in modo che restituisca sia il valore minimo sia il suo indice nella lista. Eseguire la versione modificata sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Python completo:

```

1 def array_min_value(a):
2     """Trova e restituisce il valore minimo della lista a."""
3     min_val = a[0] # Inizializza con il primo elemento
4
5     for i in range(1, len(a)):
6         if a[i] < min_val:
7             min_val = a[i] # Aggiorna il minimo
8
9     return min_val
10
11
12 if __name__ == "__main__":
13     # Caso di prova 1
14     arr1 = [9, 12, 3, 16, 19, 11, 12]
15     print(f"Prova 1 - Lista: {arr1}")
16     print(f"Valore minimo: {array_min_value(arr1)}")

```

```

17     print()
18
19     # Caso di prova 2
20     arr2 = [5, 2, 8, 1]
21     print(f"Prova 2 - Lista: {arr2}")
22     print(f"Valore minimo: {array_min_value(arr2)}")
23     print()
24
25     # Caso di prova 3
26     arr3 = [42]
27     print(f"Prova 3 - Lista: {arr3}")
28     print(f"Valore minimo: {array_min_value(arr3)}")

```

Per l'Istanza 3 il ciclo `for i in range(1, 1)` non esegue nessuna iterazione, e la funzione restituisce l'unico elemento `a[0]`.

2. **Esecuzione.** Se il programma è salvato nel file `min_value.py`, lo si esegue con il comando

```
python3 min_value.py
```

Output:

```
Prova 1 - Lista: [9, 12, 3, 16, 19, 11, 12]
Valore minimo: 3
```

```
Prova 2 - Lista: [5, 2, 8, 1]
Valore minimo: 1
```

```
Prova 3 - Lista: [42]
Valore minimo: 42
```

3. **Funzione modificata che restituisce il valore minimo e il suo indice:**

```

1 def array_min_value_with_index(a):
2     """Restituisce (valore_minimo, indice) dell'elemento minimo della
3     lista a."""
4     min_val = a[0]
5     idx = 0
6
7     for i in range(1, len(a)):
8         if a[i] < min_val:
9             min_val = a[i]
10            idx = i # Memorizza l'indice
11
12     return min_val, idx

```

```

13
14 if __name__ == "__main__":
15     arr1 = [9, 12, 3, 16, 19, 11, 12]
16     min_val, idx = array_min_value_with_index(arr1)
17     print(f"Valore minimo: {min_val}, all'indice: {idx}")

```

Output:

Valore minimo: 3, all'indice: 2

Per l'Istanza 1 la funzione restituisce la tupla (3, 2): il valore minimo 3 è all'indice 2 della lista (indici da 0), cioè nella posizione 3 dell'array $a[1..7]$ del testo. Se il valore minimo compare più volte, la funzione restituisce l'indice della sua prima occorrenza, perché `min_val` e `idx` sono aggiornati solo quando $a[i] < \text{min_val}$ (disuguaglianza stretta).

Esercizio 7.3 — Introduzione a Python (E3)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di calcolare la somma e la media di tutti i suoi elementi.

Domande

1. Scrivere un programma Python completo che calcola la somma e la media degli elementi di una lista a di n numeri interi. Il programma deve:
 - definire una funzione `array_sum(a)` che restituisce la somma di tutti gli elementi;
 - definire una funzione `array_average(a)` che restituisce la media di tutti gli elementi;
 - nel blocco `__main__`, creare le liste di test, chiamare entrambe le funzioni e stampare i risultati, con la media arrotondata a 4 cifre decimali.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare il comando che esegue il programma e l'output per tutti i casi di test.

3. Riunire le due funzioni in un'unica funzione `array_sum_and_average(a)` che restituisce sia la somma sia la media come una tupla. Eseguirla sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Python completo:

```

1 def array_sum(a):
2     """Calcola e restituisce la somma di tutti gli elementi della lista a.
3     """
4     total = 0
5     for x in a:
6         total += x
7     return total
8
9 def array_average(a):
10    """Calcola e restituisce la media di tutti gli elementi della lista a.
11    """
12    return array_sum(a) / len(a)
13
14 if __name__ == "__main__":
15     # Caso di prova 1
16     arr1 = [9, 12, 3, 16, 19, 11, 12]
17     print(f"Prova 1 - Lista: {arr1}")
18     print(f"Somma: {array_sum(arr1)}")
19     print(f"Media: {array_average(arr1):.4f}")
20     print()
21
22     # Caso di prova 2
23     arr2 = [5, 2, 8, 1]
24     print(f"Prova 2 - Lista: {arr2}")
25     print(f"Somma: {array_sum(arr2)}")
26     print(f"Media: {array_average(arr2):.4f}")
27     print()
28
29     # Caso di prova 3
30     arr3 = [42]
31     print(f"Prova 3 - Lista: {arr3}")
32     print(f"Somma: {array_sum(arr3)}")
33     print(f"Media: {array_average(arr3):.4f}")

```

Il ciclo `for x in a` visita direttamente gli elementi della lista. L'operatore `/` restituisce sempre un `float`, quindi non serve alcuna conversione di tipo (in C++ la somma deve essere convertita in `double` prima della divisione). La funzione `array_average` richiede $n \geq 1$: per una lista vuota la divisione per `len(a) = 0` ferma il programma con una `ZeroDivisionError`.

2. **Esecuzione.** Se il programma è salvato nel file `array_sum.py`, lo si esegue con il comando

```
python3 array_sum.py
```

Output:

Prova 1 - Lista: [9, 12, 3, 16, 19, 11, 12]
 Somma: 82
 Media: 11.7143

Prova 2 - Lista: [5, 2, 8, 1]
 Somma: 16
 Media: 4.0000

Prova 3 - Lista: [42]
 Somma: 42
 Media: 42.0000

Per l'Istanza 1 la media esatta è $82/7 = 11.714285\dots$, visualizzata come 11.7143.

3. Funzione combinata:

```

1 def array_sum_and_average(a):
2     """Restituisce (somma, media) della lista a."""
3     total = 0
4     for x in a:
5         total += x
6     avg = total / len(a)
7     return total, avg
8
9
10 if __name__ == "__main__":
11     arr1 = [9, 12, 3, 16, 19, 11, 12]
12     s, avg = array_sum_and_average(arr1)
13     print(f"Somma: {s}, Media: {avg:.4f}")

```

Output:

Somma: 82, Media: 11.7143

La funzione combinata visita la lista una sola volta, mentre `array_average` del primo programma chiama `array_sum` e visita la lista una seconda volta.

Esercizio 7.4 — Introduzione a Python (E4)

Dato un array a di n numeri interi e un valore m , consideriamo il problema di determinare se il valore m è presente o no nell'array.

Domande

1. Scrivere un programma Python completo che esegue una **ricerca lineare** in una lista
 - a. Il programma deve:
 - definire una funzione `linear_search(a, m)` che cerca il valore `m` nella lista `a`;
 - restituire l'indice della prima occorrenza di `m` se viene trovato, oppure `-1` se non viene trovato;
 - nel blocco `__main__`, eseguire i casi di test e stampare i risultati.
2. Eseguire il programma sulla lista `a = [9, 12, 3, 16, 19, 11, 12]` con i seguenti valori:
 - `m = 16` (presente nella lista)
 - `m = 12` (presente due volte: si restituisce l'indice della prima occorrenza)
 - `m = 7` (non presente nella lista)

Mostrare il comando che esegue il programma e l'output per tutti i casi di test.
3. Modificare la funzione in modo che restituisca la **lista di tutti gli indici** in cui compare il valore `m` (invece della sola prima occorrenza). Eseguita con `m = 12` e mostrare l'output.

Soluzione

1. Programma Python completo:

```

1 def linear_search(a, m):
2     """Cerca il valore m nella lista a.
3     Restituisce l'indice della prima occorrenza, oppure -1 se non è
4     presente."""
5     for i in range(len(a)):
6         if a[i] == m:
7             return i # Trovato: restituisce l'indice
8             return -1 # Non trovato
9
10 def print_result(a, m):
11     """Cerca m in a e stampa il risultato."""
12     idx = linear_search(a, m)
13     if idx != -1:
14         print(f"Valore {m} trovato all'indice {idx}")
15     else:
16         print(f"Valore {m} non trovato")
17
18
19 if __name__ == "__main__":
20     a = [9, 12, 3, 16, 19, 11, 12]
21     print(f"Lista: {a}")
22     print()

```

```

23
24     print_result(a, 16)
25     print_result(a, 12)
26     print_result(a, 7)

```

L'istruzione `return i` termina la funzione, e quindi il ciclo, alla prima occorrenza di `m`; l'istruzione `return -1` viene raggiunta solo quando il ciclo ha visitato l'intera lista senza trovare `m`. Si usa il valore `-1` perché non può essere un indice della lista. La funzione ausiliaria `print_result` evita di scrivere tre volte le stesse istruzioni.

2. **Esecuzione.** Se il programma è salvato nel file `linear_search.py`, lo si esegue con il comando

```
python3 linear_search.py
```

Output:

Lista: [9, 12, 3, 16, 19, 11, 12]

Valore 16 trovato all'indice 3

Valore 12 trovato all'indice 1

Valore 7 non trovato

Il valore 12 si trova negli indici 1 e 6, e la funzione restituisce il primo. Nell'array `a[1..7]` del testo, gli indici 3 e 1 corrispondono alle posizioni 4 e 2.

3. **Funzione che restituisce tutte le occorrenze:**

```

1 def linear_search_all(a, m):
2     """Restituisce la lista di tutti gli indici in cui il valore m compare
3     in a.
4     Restituisce una lista vuota se m non è presente."""
5     indices = []
6     for i in range(len(a)):
7         if a[i] == m:
8             indices.append(i)
9     return indices
10
11 if __name__ == "__main__":
12     a = [9, 12, 3, 16, 19, 11, 12]
13
14     m = 12
15     result = linear_search_all(a, m)
16     if result: # Una lista non vuota è vera, una lista vuota è falsa
17         print(f"Valore {m} trovato agli indici: {result}")
18     else:
19         print(f"Valore {m} non trovato")

```

Output:

Valore 12 trovato agli indici: [1, 6]

A differenza di `linear_search`, questa funzione visita sempre l'intera lista, perché non può fermarsi alla prima occorrenza. La lista `indices` parte vuota e cresce con `append`; quando è usata come condizione, una lista vuota vale `False`.

Esercizio 7.5 — Introduzione a Python (E5)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di calcolare in un solo passaggio il massimo, il minimo, la somma e la media di tutti gli elementi dell'array.

Domande

1. Scrivere un programma Python completo che calcola in un **solo passaggio** le quattro statistiche (massimo, minimo, somma, media) di una lista a di n numeri interi. Il programma deve:
 - definire una funzione `array_stats(a)` che calcola i quattro valori con un solo ciclo;
 - restituire i quattro valori come una tupla (`max_val`, `min_val`, `total`, `avg`);
 - nel blocco `__main__`, eseguire i casi di test e stampare i risultati.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare il comando che esegue il programma e l'output per tutti i casi di test.

3. Modificare la funzione in modo che restituisca anche gli indici dell'elemento massimo e dell'elemento minimo. Eseguirla sull'Istanza 1 e mostrare l'output.

Soluzione**1. Programma Python completo:**

```

1 def array_stats(a):
2     """Calcola massimo, minimo, somma e media in un solo passaggio.
3     Restituisce (max_val, min_val, total, avg)."""
4     max_val = a[0]
```

```

5     min_val = a[0]
6     total = a[0]
7
8     for i in range(1, len(a)):
9         if a[i] > max_val:
10            max_val = a[i]
11        if a[i] < min_val:
12            min_val = a[i]
13        total += a[i]
14
15    avg = total / len(a)
16    return max_val, min_val, total, avg
17
18
19 def print_stats(k, a):
20     """Stampa le statistiche del caso di prova k."""
21     max_v, min_v, s, avg = array_stats(a)
22     print(f"Prova {k} - Lista: {a}")
23     print(f" Massimo: {max_v}")
24     print(f" Minimo:  {min_v}")
25     print(f" Somma:   {s}")
26     print(f" Media:    {avg:.4f}")
27
28
29 if __name__ == "__main__":
30     print_stats(1, [9, 12, 3, 16, 19, 11, 12])
31     print()
32     print_stats(2, [5, 2, 8, 1])
33     print()
34     print_stats(3, [42])

```

Le tre variabili sono inizializzate con il primo elemento, e il ciclo visita una sola volta gli altri $n - 1$ elementi: a ogni iterazione l'elemento corrente è confrontato con il massimo e con il minimo, e viene sommato alla somma.

2. **Esecuzione.** Se il programma è salvato nel file `array_stats.py`, lo si esegue con il comando

```
python3 array_stats.py
```

Output:

```

Prova 1 - Lista: [9, 12, 3, 16, 19, 11, 12]
Massimo: 19
Minimo:  3
Somma:   82
Media:   11.7143

Prova 2 - Lista: [5, 2, 8, 1]

```

```

Massimo: 8
Minimo:  1
Somma:   16
Media:   4.0000

```

Prova 3 - Lista: [42]

```

Massimo: 42
Minimo:  42
Somma:   42
Media:   42.0000

```

3. Funzione estesa che restituisce anche gli indici:

```

1 def array_stats_with_indices(a):
2     """Calcola tutte le statistiche, comprese le posizioni.
3     Restituisce (max_val, max_idx, min_val, min_idx, total, avg)."""
4     max_val = a[0]
5     max_idx = 0
6     min_val = a[0]
7     min_idx = 0
8     total = a[0]
9
10    for i in range(1, len(a)):
11        if a[i] > max_val:
12            max_val = a[i]
13            max_idx = i
14        if a[i] < min_val:
15            min_val = a[i]
16            min_idx = i
17        total += a[i]
18
19    avg = total / len(a)
20    return max_val, max_idx, min_val, min_idx, total, avg
21
22
23 if __name__ == "__main__":
24     a = [9, 12, 3, 16, 19, 11, 12]
25     max_v, max_i, min_v, min_i, s, avg = array_stats_with_indices(a)
26     print(f"Lista:   {a}")
27     print(f"Max:     {max_v} all'indice {max_i}")
28     print(f"Min:     {min_v} all'indice {min_i}")
29     print(f"Somma:    {s}")
30     print(f"Media:    {avg:.4f}")

```

Output:

```

Lista:   [9, 12, 3, 16, 19, 11, 12]
Max:     19 all'indice 4
Min:     3 all'indice 2
Somma:   82
Media:   11.7143

```

Il massimo 19 è all'indice 4 e il minimo 3 all'indice 2 (indici da 0), cioè nelle posizioni 5 e 3 dell'array $a[1..7]$ del testo.

Introduzione a Java

Negli esercizi di questo capitolo i programmi sono scritti, compilati ed eseguiti con le convenzioni del capitolo «Introduzione a Java», riassunte qui sotto.

Attenzione

- Il programma di un esercizio è una classe **Main**, salvata nel file **Main.java** (ogni esercizio in una cartella propria), compilata ed eseguita con i comandi

```
javac Main.java
java Main
```

- I metodi sono metodi **static** della classe **Main**. Le istanze di test sono array scritti nel codice, ad esempio `int[] arr1 = {9, 12, 3, 16, 19, 11, 12};`, e sono passate ai metodi senza la loro dimensione: un metodo ottiene la dimensione n di un array **a** con `a.length`.
- Gli elementi di un array di dimensione n sono `a[0]`, `a[1]`, ..., `a[n-1]`: la componente `a[i]` di un'istanza scritta come `a = [...]` nel testo corrisponde ad `a[i-1]` nel codice, e gli indici stampati dai programmi partono da 0.
- Un metodo restituisce al più un valore: i risultati che un metodo deve restituire insieme sono restituiti in un array (`int[]` o `double[]`).
- Quando un array è stampato con un ciclo, ogni valore è seguito da uno spazio, quindi le righe stampate terminano con uno spazio.
- Il formato `%.4f` di `printf` visualizza un numero in virgola mobile con esattamente 4 cifre dopo la virgola decimale. Gli output sono quelli di un calcolatore con impostazioni in inglese: con le impostazioni in italiano `printf` scrive la virgola decimale (11,7143).

Esercizio 8.1 — Introduzione a Java (E1)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di trovare il valore massimo nell'array.

Domande

1. Scrivere un programma Java completo che trova il valore massimo in un array a di n numeri interi. Il programma deve:
 - definire un metodo statico `int arrayMaxValue(int[] a)` dentro la classe `Main`;
 - nel metodo `main`, creare gli array di test e chiamare il metodo;
 - stampare i risultati per ogni caso di test.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare i comandi di compilazione ed esecuzione e l'output per tutti i casi di test.

3. Modificare il metodo in modo che restituisca sia il valore massimo sia il suo indice. Poiché un metodo Java restituisce un solo valore, restituire un array `int[]` di due elementi: `{maxValue, maxIndex}`. Eseguire la versione modificata sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Java completo:

```

1 public class Main {
2
3     static int arrayMaxValue(int[] a) {
4         int max = a[0]; // Inizializza con il primo elemento
5
6         for (int i = 1; i < a.length; i++) {
7             if (a[i] > max) {
8                 max = a[i]; // Aggiorna il massimo
9             }
10        }
11        return max;
12    }
13
14    public static void main(String[] args) {
15        int[] arr1 = {9, 12, 3, 16, 19, 11, 12};
16        System.out.print("Prova 1 - Array: ");
17        for (int x : arr1) System.out.print(x + " ");
18        System.out.println();
19        System.out.println("Valore massimo: " + arrayMaxValue(arr1));
20        System.out.println();
21
22        int[] arr2 = {5, 2, 8, 1};
23        System.out.print("Prova 2 - Array: ");
24        for (int x : arr2) System.out.print(x + " ");

```

```

25     System.out.println();
26     System.out.println("Valore massimo: " + arrayMaxValue(arr2));
27     System.out.println();
28
29     int[] arr3 = {42};
30     System.out.print("Prova 3 - Array: ");
31     for (int x : arr3) System.out.print(x + " ");
32     System.out.println();
33     System.out.println("Valore massimo: " + arrayMaxValue(arr3));
34 }
35 }

```

Gli array sono stampati con il ciclo `for` esteso, che visita direttamente i loro elementi. Per l'Istanza 3 la condizione `1 < a.length` è falsa fin dall'inizio, il ciclo non esegue nessuna iterazione, e il metodo restituisce l'unico elemento `a[0]`.

2. Compilazione ed esecuzione:

```

javac Main.java
java Main

```

Output:

```

Prova 1 - Array: 9 12 3 16 19 11 12
Valore massimo: 19

Prova 2 - Array: 5 2 8 1
Valore massimo: 8

Prova 3 - Array: 42
Valore massimo: 42

```

3. Metodo modificato che restituisce il valore massimo e il suo indice:

```

1 public class Main {
2
3     // Restituisce int[]{maxValue, maxIndex}
4     static int[] arrayMaxValueWithIndex(int[] a) {
5         int max = a[0];
6         int idx = 0;
7
8         for (int i = 1; i < a.length; i++) {
9             if (a[i] > max) {
10                 max = a[i];
11                 idx = i; // Memorizza l'indice
12             }
13         }
14         return new int[]{max, idx};
15     }
16 }

```

```

16
17     public static void main(String[] args) {
18         int[] arr1 = {9, 12, 3, 16, 19, 11, 12};
19         int[] result = arrayMaxValueWithIndex(arr1);
20         System.out.println("Valore massimo: " + result[0]
21                             + ", all'indice: " + result[1]);
22     }
23 }

```

Output:

Valore massimo: 19, all'indice: 4

Per l'Istanza 1 il metodo restituisce l'array {19, 4}: il valore massimo 19 è all'indice 4 (indici da 0), cioè nella posizione 5 dell'array $a[1..7]$ del testo. Se il valore massimo compare più volte, il metodo restituisce l'indice della sua prima occorrenza, perché $\max$ e idx sono aggiornati solo quando $a[i] > \max$ (disuguaglianza stretta).

Esercizio 8.2 — Introduzione a Java (E2)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di trovare il valore minimo nell'array.

Domande

1. Scrivere un programma **Java** completo che trova il valore minimo in un array a di n numeri interi. Il programma deve:
 - definire un metodo statico `int arrayMinValue(int[] a)` dentro la classe `Main`;
 - nel metodo `main`, creare gli array di test e chiamare il metodo;
 - stampare i risultati per ogni caso di test.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare i comandi di compilazione ed esecuzione e l'output per tutti i casi di test.

3. Modificare il metodo in modo che restituisca sia il valore minimo sia il suo indice. Poiché un metodo **Java** restituisce un solo valore, restituire un array `int[]` di due elementi: `{minValue, minIndex}`. Eseguire la versione modificata sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Java completo:

```
1 public class Main {
2
3     static int arrayMinValue(int[] a) {
4         int min = a[0]; // Inizializza con il primo elemento
5
6         for (int i = 1; i < a.length; i++) {
7             if (a[i] < min) {
8                 min = a[i]; // Aggiorna il minimo
9             }
10        }
11        return min;
12    }
13
14    public static void main(String[] args) {
15        int[] arr1 = {9, 12, 3, 16, 19, 11, 12};
16        System.out.print("Prova 1 - Array: ");
17        for (int x : arr1) System.out.print(x + " ");
18        System.out.println();
19        System.out.println("Valore minimo: " + arrayMinValue(arr1));
20        System.out.println();
21
22        int[] arr2 = {5, 2, 8, 1};
23        System.out.print("Prova 2 - Array: ");
24        for (int x : arr2) System.out.print(x + " ");
25        System.out.println();
26        System.out.println("Valore minimo: " + arrayMinValue(arr2));
27        System.out.println();
28
29        int[] arr3 = {42};
30        System.out.print("Prova 3 - Array: ");
31        for (int x : arr3) System.out.print(x + " ");
32        System.out.println();
33        System.out.println("Valore minimo: " + arrayMinValue(arr3));
34    }
35 }
```

Gli array sono stampati con il ciclo `for` esteso, che visita direttamente i loro elementi. Per l'Istanza 3 la condizione `1 < a.length` è falsa fin dall'inizio, il ciclo non esegue nessuna iterazione, e il metodo restituisce l'unico elemento `a[0]`.

2. Compilazione ed esecuzione:

```
javac Main.java
java Main
```

Output:

```
Prova 1 - Array: 9 12 3 16 19 11 12
Valore minimo: 3
```

Prova 2 - Array: 5 2 8 1
Valore minimo: 1

Prova 3 - Array: 42
Valore minimo: 42

3. Metodo modificato che restituisce il valore minimo e il suo indice:

```

1 public class Main {
2
3     // Restituisce int[]{minValue, minIndex}
4     static int[] arrayMinValueWithIndex(int[] a) {
5         int min = a[0];
6         int idx = 0;
7
8         for (int i = 1; i < a.length; i++) {
9             if (a[i] < min) {
10                 min = a[i];
11                 idx = i; // Memorizza l'indice
12             }
13         }
14         return new int[]{min, idx};
15     }
16
17     public static void main(String[] args) {
18         int[] arr1 = {9, 12, 3, 16, 19, 11, 12};
19         int[] result = arrayMinValueWithIndex(arr1);
20         System.out.println("Valore minimo: " + result[0]
21                             + ", all'indice: " + result[1]);
22     }
23 }

```

Output:

Valore minimo: 3, all'indice: 2

Per l'Istanza 1 il metodo restituisce l'array {3, 2}: il valore minimo 3 è all'indice 2 (indici da 0), cioè nella posizione 3 dell'array $a[1..7]$ del testo. Se il valore minimo compare più volte, il metodo restituisce l'indice della sua prima occorrenza, perché min e idx sono aggiornati solo quando $a[i] < \text{min}$ (disuguaglianza stretta).

Esercizio 8.3 — Introduzione a Java (E3)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di calcolare la somma e la media di tutti i suoi elementi.

Domande

1. Scrivere un programma Java completo che calcola la somma e la media degli elementi di un array a di n numeri interi. Il programma deve:
 - definire un metodo statico `int arraySum(int[] a)` che restituisce la somma;
 - definire un metodo statico `double arrayAverage(int[] a)` che restituisce la media;
 - nel metodo `main`, creare gli array di test, chiamare entrambi i metodi e stampare i risultati, con la media arrotondata a 4 cifre decimali.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare i comandi di compilazione ed esecuzione e l'output per tutti i casi di test.

3. Scrivere un unico metodo `arraySumAndAverage(int[] a)` che restituisce sia la somma sia la media. Poiché un metodo Java restituisce un solo valore, restituire il risultato come un array `double[]` di due elementi: `{sum, average}`. Eseguirlo sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Java completo:

```

1 public class Main {
2
3     static int arraySum(int[] a) {
4         int total = 0;
5         for (int i = 0; i < a.length; i++) {
6             total += a[i];
7         }
8         return total;
9     }
10
11     static double arrayAverage(int[] a) {
12         return (double) arraySum(a) / a.length;
13     }
14
15     static void printTest(int k, int[] a) {
16         System.out.print("Prova " + k + " - Array: ");
17         for (int x : a) System.out.print(x + " ");
18         System.out.println();
19         System.out.println("Somma: " + arraySum(a));
20         System.out.printf("Media: %.4f%n", arrayAverage(a));
21     }
22
23     public static void main(String[] args) {

```

```

24     int[] arr1 = {9, 12, 3, 16, 19, 11, 12};
25     printTest(1, arr1);
26     System.out.println();
27
28     int[] arr2 = {5, 2, 8, 1};
29     printTest(2, arr2);
30     System.out.println();
31
32     int[] arr3 = {42};
33     printTest(3, arr3);
34 }
35 }

```

La conversione (`double`) è necessaria: senza di essa, `arraySum(a) / a.length` sarebbe la divisione intera, e per l'Istanza 1 la media sarebbe $82/7 = 11$. Il metodo `printTest` stampa l'array e i due risultati, ed evita di scrivere tre volte le stesse istruzioni.

2. Compilazione ed esecuzione:

```

javac Main.java
java Main

```

Output:

```

Prova 1 - Array: 9 12 3 16 19 11 12
Somma: 82
Media: 11.7143

```

```

Prova 2 - Array: 5 2 8 1
Somma: 16
Media: 4.0000

```

```

Prova 3 - Array: 42
Somma: 42
Media: 42.0000

```

Per l'Istanza 1 la media esatta è $82/7 = 11.714285\dots$, visualizzata come 11.7143.

3. Metodo combinato:

```

1 public class Main {
2
3     // Restituisce double[]{somma, media}
4     static double[] arraySumAndAverage(int[] a) {
5         int total = 0;
6         for (int x : a) total += x;
7         double avg = (double) total / a.length;
8         return new double[]{total, avg};
9     }
10 }

```

```

9      }
10
11     public static void main(String[] args) {
12         int[] arr1 = {9, 12, 3, 16, 19, 11, 12};
13         double[] result = arraySumAndAverage(arr1);
14         System.out.printf("Somma: %.0f, Media: %.4f\n",
15                             result[0], result[1]);
16     }
17 }

```

Output:

Somma: 82, Media: 11.7143

L'intero `total` è convertito automaticamente in un `double` quando viene memorizzato nell'array `double[]`; per questo motivo è stampato con il formato `%.0f` (nessuna cifra decimale) invece di `%d`, che accetta solo interi.

Esercizio 8.4 — Introduzione a Java (E4)

Dato un array a di n numeri interi e un valore m , consideriamo il problema di determinare se il valore m è presente o no nell'array.

Domande

1. Scrivere un programma Java completo che esegue una **ricerca lineare** in un array a . Il programma deve:
 - definire un metodo statico `int linearSearch(int[] a, int m)` che cerca il valore m nell'array a ;
 - restituire l'indice della prima occorrenza di m se viene trovato, oppure `-1` se non viene trovato;
 - nel metodo `main`, eseguire i casi di test e stampare i risultati.
2. Eseguire il programma sull'array $a = [9, 12, 3, 16, 19, 11, 12]$ con i seguenti valori:
 - $m = 16$ (presente nell'array)
 - $m = 12$ (presente due volte: si restituisce l'indice della prima occorrenza)
 - $m = 7$ (non presente nell'array)

Mostrare i comandi di compilazione ed esecuzione e l'output per tutti i casi di test.

3. Modificare il metodo in modo che restituisca un array `int[]` contenente **tutti gli indici** in cui compare il valore m (invece della sola prima occorrenza). Usare una `ArrayList<Integer>` dentro il metodo e convertirla in un `int[]` prima di restituirla. Eseguirlo con $m = 12$ e mostrare l'output.

Soluzione

1. Programma Java completo:

```

1 public class Main {
2
3     static int linearSearch(int[] a, int m) {
4         for (int i = 0; i < a.length; i++) {
5             if (a[i] == m) {
6                 return i; // Trovato: restituisce l'indice
7             }
8         }
9         return -1; // Non trovato
10    }
11
12    static void printResult(int[] a, int m) {
13        int idx = linearSearch(a, m);
14        if (idx != -1) System.out.println("Valore " + m + " trovato all'indice " +
15            idx);
16        else
17            System.out.println("Valore " + m + " non trovato");
18    }
19
20    public static void main(String[] args) {
21        int[] a = {9, 12, 3, 16, 19, 11, 12};
22        System.out.print("Array: ");
23        for (int x : a) System.out.print(x + " ");
24        System.out.println();
25        System.out.println();
26
27        printResult(a, 16);
28        printResult(a, 12);
29        printResult(a, 7);
30    }
31 }

```

L'istruzione `return i` termina il metodo, e quindi il ciclo, alla prima occorrenza di `m`; l'istruzione `return -1` viene raggiunta solo quando il ciclo ha visitato l'intero array senza trovare `m`. Si usa il valore `-1` perché non può essere un indice dell'array.

2. Compilazione ed esecuzione:

```

javac Main.java
java Main

```

Output:

```
Array: 9 12 3 16 19 11 12
```

```

Valore 16 trovato all'indice 3
Valore 12 trovato all'indice 1
Valore 7 non trovato

```

Il valore 12 si trova negli indici 1 e 6, e il metodo restituisce il primo. Nell'array $a[1..7]$ del testo, gli indici 3 e 1 corrispondono alle posizioni 4 e 2.

3. Metodo che restituisce tutte le occorrenze:

```

1  import java.util.ArrayList;
2
3  public class Main {
4
5      static int[] linearSearchAll(int[] a, int m) {
6          ArrayList<Integer> indices = new ArrayList<>();
7          for (int i = 0; i < a.length; i++) {
8              if (a[i] == m) {
9                  indices.add(i);
10             }
11         }
12         // Conversione della ArrayList<Integer> in un int[]
13         int[] result = new int[indices.size()];
14         for (int i = 0; i < indices.size(); i++) {
15             result[i] = indices.get(i);
16         }
17         return result;
18     }
19
20     public static void main(String[] args) {
21         int[] a = {9, 12, 3, 16, 19, 11, 12};
22         int m = 12;
23         int[] occurrences = linearSearchAll(a, m);
24
25         if (occurrences.length > 0) {
26             System.out.print("Valore " + m + " trovato agli indici: ");
27             for (int idx : occurrences) System.out.print(idx + " ");
28             System.out.println();
29         } else {
30             System.out.println("Valore " + m + " non trovato");
31         }
32     }
33 }

```

Output:

Valore 12 trovato agli indici: 1 6

Il numero di occorrenze non è noto in anticipo, quindi gli indici sono prima raccolti in una `ArrayList`, che cresce con `add`; poi si alloca e si riempie un array della dimensione giusta `indices.size()`. A differenza di `linearSearch`, questo metodo visita sempre l'intero array. Se m non è presente, il metodo restituisce un array di lunghezza 0.

Esercizio 8.5 — Introduzione a Java (E5)

Dato un array a di $n \geq 1$ numeri interi, consideriamo il problema di calcolare in un solo passaggio il massimo, il minimo, la somma e la media di tutti gli elementi dell'array.

Domande

1. Scrivere un programma Java completo che calcola in un **solo passaggio** le quattro statistiche (massimo, minimo, somma, media) di un array a di n numeri interi. Il programma deve:
 - definire un metodo statico `arrayStats(int[] a)` che calcola i quattro valori con un solo ciclo;
 - restituire i risultati come un array `double[]` di quattro elementi: {max, min, sum, average};
 - nel metodo `main`, eseguire i casi di test e stampare i risultati.
2. Eseguire il programma sulle seguenti istanze:
 - Istanza 1: $a = [9, 12, 3, 16, 19, 11, 12]$ con $n = 7$
 - Istanza 2: $a = [5, 2, 8, 1]$ con $n = 4$
 - Istanza 3: $a = [42]$ con $n = 1$

Mostrare i comandi di compilazione ed esecuzione e l'output per tutti i casi di test.

3. Modificare il metodo in modo che restituisca anche gli indici dell'elemento massimo e dell'elemento minimo, come un array `double[]` di sei elementi: {max, maxIdx, min, minIdx, sum, average}. Eseguirlo sull'Istanza 1 e mostrare l'output.

Soluzione

1. Programma Java completo:

```

1 public class Main {
2
3     // Restituisce double[]{massimo, minimo, somma, media}
4     static double[] arrayStats(int[] a) {
5         int max = a[0];
6         int min = a[0];
7         int total = a[0];
8
9         for (int i = 1; i < a.length; i++) {
10             if (a[i] > max) max = a[i];
11             if (a[i] < min) min = a[i];
12             total += a[i];
13         }

```

```

14
15     double avg = (double) total / a.length;
16     return new double[]{max, min, total, avg};
17 }
18
19 static void printStats(int k, int[] a) {
20     double[] s = arrayStats(a);
21     System.out.print("Prova " + k + " - Array: ");
22     for (int x : a) System.out.print(x + " ");
23     System.out.println();
24     System.out.println(" Massimo: " + (int) s[0]);
25     System.out.println(" Minimo:   " + (int) s[1]);
26     System.out.println(" Somma:   " + (int) s[2]);
27     System.out.printf(" Media:    %.4f%n", s[3]);
28 }
29
30 public static void main(String[] args) {
31     printStats(1, new int[]{9, 12, 3, 16, 19, 11, 12});
32     System.out.println();
33     printStats(2, new int[]{5, 2, 8, 1});
34     System.out.println();
35     printStats(3, new int[]{42});
36 }
37 }

```

Le tre variabili sono inizializzate con il primo elemento, e il ciclo visita una sola volta gli altri $n - 1$ elementi. I quattro risultati hanno tipi di dato diversi, quindi sono tutti memorizzati come `double` nell'array restituito; i risultati interi sono riconvertiti con `(int)` prima della stampa, in modo che siano visualizzati senza la parte decimale (19 invece di 19.0). L'espressione `new int[]{...}` crea un array direttamente nella chiamata.

2. Compilazione ed esecuzione:

```

javac Main.java
java Main

```

Output:

```

Prova 1 - Array: 9 12 3 16 19 11 12
Massimo: 19
Minimo:  3
Somma:   82
Media:   11.7143

Prova 2 - Array: 5 2 8 1
Massimo: 8
Minimo:  1
Somma:   16
Media:   4.0000

```

```
Prova 3 - Array: 42
Massimo: 42
Minimo: 42
Somma: 42
Media: 42.0000
```

3. Metodo esteso che restituisce anche gli indici:

```
1 public class Main {
2
3     // Restituisce double[]{max, maxIdx, min, minIdx, somma, media}
4     static double[] arrayStatsWithIndices(int[] a) {
5         int max = a[0], maxIdx = 0;
6         int min = a[0], minIdx = 0;
7         int total = a[0];
8
9         for (int i = 1; i < a.length; i++) {
10             if (a[i] > max) { max = a[i]; maxIdx = i; }
11             if (a[i] < min) { min = a[i]; minIdx = i; }
12             total += a[i];
13         }
14
15         double avg = (double) total / a.length;
16         return new double[]{max, maxIdx, min, minIdx, total, avg};
17     }
18
19     public static void main(String[] args) {
20         int[] a = {9, 12, 3, 16, 19, 11, 12};
21         double[] r = arrayStatsWithIndices(a);
22         System.out.println("Max:   " + (int) r[0] + " all'indice " + (int) r[1]);
23         System.out.println("Min:   " + (int) r[2] + " all'indice " + (int) r[3]);
24         System.out.println("Somma: " + (int) r[4]);
25         System.out.printf("Media: %.4f%n", r[5]);
26     }
27 }
```

Output:

```
Max:   19 all'indice 4
Min:   3 all'indice 2
Somma: 82
Media: 11.7143
```

Il massimo 19 è all'indice 4 e il minimo 3 all'indice 2 (indici da 0), cioè nelle posizioni 5 e 3 dell'array `a[1..7]` del testo.