



Insegnamento di Programmazione (6 CFU)

Esercizi Parte 3: alcune soluzioni

Ing. Nadia Ranaldo
Dipartimento di Ingegneria
Università degli Studi del Sannio

Parte 3. Esercizio 1



- Scrivere un programma C che stabilisce se un array di N interi letti da tastiera è palindromo, ovvero se gli elementi presi dal primo all'ultimo sono uguali a quelli letti dall'ultimo al primo.

Esempio

[1 2 3 2 1] è palindromo
[1 2 3 3 2 1] è palindromo
[1 2 3 1] non è palindromo

2

Soluzione (1)



Struttura del programma

- **main**
 - Lettura degli interi da tastiera ed inserimento in un array di N elementi (funzione `inputArray`)
- `int inputArray(int a[], int cap)`
- Determinazione se l'array è palindromo (funzione predicato `palindromo`)
- `int palindromo(int a[], int riemp)`
 - Funzione che restituisce un valore booleano TRUE se l'array è palindromo, FALSE altrimenti

Es. [35, 18, 2, 17, 2, 18, 35]



3

Soluzione (1)



- Occorre confrontare coppie di elementi
 - (primo, ultimo), (secondo, penultimo) etc.
 - Appena si trova una coppia di elementi diversi, l'array non è palindromo, altrimenti si procede con l'analizzare un'altra coppia di elementi
- Quante coppie di elementi devono essere analizzate? Numero degli elementi/2 (`riemp/2`)
 - Si ottengono in generale le coppie date dagli indici (`i, riemp-1-i`), con `i` che va da 0 a `riemp/2` (divisione intera)
 - Se `riemp` è dispari, resta un unico elemento e quindi non occorre prenderlo in considerazione

4

Soluzione (1)



```
#include<stdio.h>
#define N 5
#define TRUE 1
#define FALSE 0
int inputArray(int a[], int c);
int palindromo(int a[], int riemp);

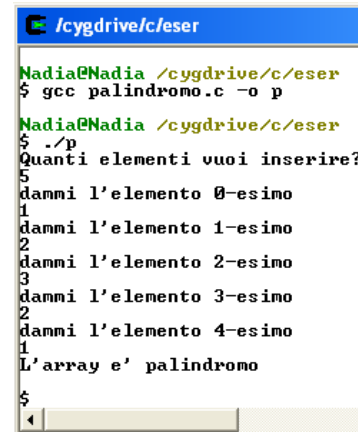
main() {
    int a[N],r,p;
    r = inputArray(a,N);
    p= palindromo(a,r);
    if (p)
        printf("L'array e' palindromo\n");
    else
        printf("L'array non e' palindromo\n");
}
```

5

Soluzione (1)



```
int inputArray(int a[], int c){
    int i, r=0;
    printf("Quanti elementi vuoi inserire? (max %d)\n",N);
    scanf("%d",&r);
    for (i=0; i<r; i++) {
        printf("dammi l'elemento %d-esimo\n", i);
        scanf("%d", &a[i]);
    }
    return r;
}
int palindromo(int a[], int riemp) {
    int i, limit;
    int palindromo=TRUE;
    limit = riemp/2;
    for (i=0; palindromo && i<limit; i++)
        palindromo=(a[i]==a[(riemp-1)-i]);
    return palindromo;
}
```



```
~/cygdrive/c/eser
Nadia@Nadia /cygdrive/c/eser
$ gcc palindromo.c -o p
Nadia@Nadia /cygdrive/c/eser
$ ./p
Quanti elementi vuoi inserire?
5
dammi l'elemento 0-esimo
1
dammi l'elemento 1-esimo
2
dammi l'elemento 2-esimo
3
dammi l'elemento 3-esimo
2
dammi l'elemento 4-esimo
1
L'array e' palindromo
$
```

Parte 3. Esercizio 2



- Scrivere una funzione `inverti` che riceve un array di interi e ne inverte l'ordine degli elementi.
- Scrivere inoltre un programma che utilizza una funzione per leggere gli interi da tastiera e la funzione `inverti`.

Esempio

Input

[3, 1, 8, 14, 12, 13, 54]

Output

[54, 13, 12, 14, 8, 1, 3]

7

Soluzione (2)



Struttura del programma

- `main`
 - Lettura degli interi da tastiera ed inserimento in un array di N elementi (funzione `inputArray`)
 - Versione della funzione per la lettura che legge un numero di interi pari alla capacità dell'array
- `void inputArray(int a[], int cap)`
- Inversione dell'array mediante la funzione `inverti`
- `void inverti(int a[], int riemp)`
- Stampa dell'array mediante la funzione `stampa`
- `void stampa(int a[], int riemp)`

8

Soluzione (2)



Algoritmo per la funzione inverti:

- Scambiare le coppie di elementi
 - (primo, ultimo), (secondo, penultimo), etc
 - Utilizzare un indice i che va da 0 a $\text{cap}/2$
 - Generica coppia individuata dagli indici $(i, (\text{cap}-1)-i)$
 - Se riemp è dispari, resta un unico elemento e quindi non occorre prenderlo in considerazione

```
[3, 1, 8, 14, 12, 13, 54]
[54, 1, 8, 14, 12, 13, 3]
[54, 13, 8, 14, 12, 1, 3]
[54, 13, 12, 14, 8, 1, 3]
```

9

Soluzione (2)



```
void inverti(int a[], int r){
    int i, temp, limit;
    limit = r/2;
    for (i=0; i<limit; i++) {
        temp = a[i];
        a[i] = a[r-1-i];
        a[r-1-i]=temp;
    }
}

void stampa(int a[], int r){
    int i;
    for (i=0; i<r; i++) {
        printf("Elemento n.%d : %d\n", i, a[i]);
    }
}
```

```

Nadia@Nadia /cygdrive/c/eser
$ gcc inverti.c -o i

Nadia@Nadia /cygdrive/c/eser
$ ./i
dammi l'el. 0-esimo
1
dammi l'el. 1-esimo
2
dammi l'el. 2-esimo
3
dammi l'el. 3-esimo
4
dammi l'el. 4-esimo
5
Elemento n.0 : 5
Elemento n.1 : 4
Elemento n.2 : 3
Elemento n.3 : 2
Elemento n.4 : 1
$
```

11

Soluzione (2)



```
#include<stdio.h>
#define N 5
void inputArray(int a[], int c);
void inverti(int a[], int r);
void stampa (int a[], int r);

main() {
    int a[N],r;
    inputArray(a,N);
    inverti(a,N);
    stampa(a,N);
}

void inputArray(int a[], int c){
    int i;
    for (i=0; i<c; i++) {
        printf("dammi l'el. %d-esimo\n", i);
        scanf("%d", &a[i]);
    }
}
```

10

Parte 3. Esercizio 5



- Un polinomio $p(x)$ di un variabile x con coefficienti reali di grado n , può essere rappresentato come un vettore di dimensione $n+1$ in cui in posizione $t[i]$ salviamo il coefficiente del termine di grado i -esimo x^i . Scrivere due funzioni che:
 - dati due polinomi, p e q calcoli il polinomio somma $p+q$;
 - dato un polinomio p calcoli il polinomio $q(x)=\delta p(x)/\delta x$ (derivata di p)
- Scrivere inoltre un programma che utilizzi una funzione per leggere i coefficienti di un polinomio da tastiera e le due funzioni sopra descritte.
- Supportare ad esempio che il grado massimo sia 9.

12

Soluzione (5)



- Esempio di rappresentazione di un polinomio $p(x)$

$$p(x) = 3x^4 + 2x^3 - x + 3$$

Indici: 4 3 2 1 0
[3, 2, 0, -1, 3]

La somma di due polinomi corrisponde al polinomio avente come grado il grado massimo tra due polinomi e come coefficienti la somma dei coefficienti dei termini di stesso grado a partire dal grado massimo fino al grado zero

Esempio

$$p(x) = 3x^4 + 2x^3 - x + 3$$

$$q(x) = 4x^5 + x^2 + 3x$$

[0, 3, 2, 0, -1, 3]
[4, 0, 0, 1, 3, 0]
[4, 3, 2, 1, -2, 3]

$$p(x) + q(x) = 4x^5 + 3x^4 + 2x^3 + x^2 - 2x + 3$$

13

Soluzione (5)



- Indicato con:

- n il grado del polinomio
- α_i il coefficiente di grado i

La derivata prima di un polinomio $p(x)$ è data da:

$$d(x) = \frac{\partial p(x)}{\partial x} = \sum_{i=1}^n (i) \alpha_i x^{i-1}$$

Esempio

$$p(x) = 3x^4 + 2x^3 - x + 3$$

$$d(x) = \delta p(x) / \delta x = 12x^3 + 6x^2 - 1$$

grado (indice) [5 4 3 2 1 0] moltiplicazione

[0 3 2 0 -1 3]

$\alpha_i = (i+1) \alpha_{i+1}$

[0, 0, 12, 6, 0, -1]

14

Soluzione (5)



Struttura del programma

- main

- Lettura di due polinomi $p(x)$ e $q(x)$ mediante la lettura dei coefficienti da tastiera ed inserimento in due array di 10 elementi (funzione `inputPolinomio`)

- Il polinomio può avere massimo un grado pari a 9
- Restituisce il numero di elementi effettivamente inseriti (riempimento dell'array=grado+1)
- Inizializza a zero i coefficienti rimanenti

`int inputPolinomio(int p[], int cap)`

- Somma dei due polinomi letti da tastiera mediante la funzione `calcola_somma`.

- I coefficienti del polinomio somma è memorizzato in un terzo array passato come parametro

`void calcola_somma(int p[], int q[], int s[], int cap)`

15

Soluzione (5)



- Stampa dell'array contenente i coefficienti del polinomio somma mediante la funzione `stampa`

`void stampa(int a[], int riemp)`

- Calcolo della derivata del polinomio $p(x)$ mediante la funzione `calcola_derivata`. I coefficienti del polinomio risultato sono memorizzati in un altro array passato come parametro.

`void calcola_derivata(int p[], int q[], int riemp)`

- Stampa più sofisticata: stampa del polinomio nella forma:

$$x^n + \dots + \alpha_1 x + \alpha_0$$

mediante la funzione `stampa_polinomio`

- Parametri di ingresso: array dei coefficienti e riempimento dell'array (grado del polinomio + 1)

`void stampa_polinomio(int p[], int riemp);`

16

Soluzione (5)



```
#include<stdio.h>
#define N 10
int inputPolinomio (int a[], int c);
void calcola_somma(int p[], int q[], int somma[], int cap);
void calcola_derivata (int p[], int der[], int r);
void stampa (int a[], int r);
void stampa_polinomio(int p[], int r);

main() {
    int p[N], q[N], s[N], der[N];
    int r1,r2;
    r1=inputPolinomio(p,N);
    printf("Stampa del polinomio p(x): \n");
    stampa(p,r1);
    stampa_polinomio(p,r1);
    r2=inputPolinomio (q,N);
    printf("Stampa del polinomio q(x): \n");
    stampa(q,r2);
    stampa_polinomio(q,r2);
    calcola_somma(p,q,s,N);
```

17

Soluzione (5)



```
printf("Stampa della somma: \n");
    stampa(s,N);
    stampapol(s,N);
    calcola_derivata(p,der,r1);
    printf("Stampa della derivata di p(x): \n");
    stampa(der,r1-1); /* derivata ha grado di p(x)-1 */
    stampapol(der,r1-1);
}
int inputPolinomio (int a[], int c) {
    int i;
    int r=0;
    printf("Di che grado è il polonomio? (massimo 9)\n");
    scanf("%d",&r);
    for (i=0; i<=r; i++) {
        printf("dammi il coefficiente di grado %d-esimo\n", i);
        scanf("%d", &a[i]);
    }
    for (i=r+1; i<c; i++) {
        a[i]=0;
    }
    return r+1;
}
```

18

Soluzione (5)



```
void calcola_derivata (int p[], int der[], int r){
    int i;
    der[r-1]=0;
    for(i=0; i<r-1; i++){
        der[i] = p[i+1]*(i+1);
    }
}
void calcola_somma(int p[], int q[], int s[], int cap){
    int i;
    for(i=0; i<cap; i++){
        s[i] = p[i]+q[i];
    }
}
void stampa(int a[], int r){
    int i;
    for (i=r-1; i>=0; i--)
        printf("%d ",a[i]);
    printf("\n");
}
```

19

Soluzione (5)



```
void stampa_polinomio(int a[], int r){
    int i, primo=1;
    for (i=r-1; i>=0; i--) {
        /* se il coeff. è nullo, occorre stampare + solo se
           non è l'ultimo, non il primo e se il coeff. succ.
           non è nullo */

        if (!a[i]) {
            if (i && !primo && a[i-1] ) printf(" + ");
            continue;
        } else primo=0;

        /* poichè c'è continue, qui a[i] è diverso da zero */
        /* il coeff. deve essere stampato se è il coeff. del
           termine di grado zero oppure se è != 1 negli
           altri casi*/

        if(!i) /* termine di grado zero */
            printf("%d",a[i]);
```

20

Soluzione (5)



```

else { /* termine di grado>0, stampare x, stampare
       inoltre il grado solo se questo è diverso da 1
       */
    if(a[i]!=1)
        printf("%d",a[i]);
    printf("x");
    if (i>1)
        printf("^%d",i);

    /* se il coeff. del grado prec. è != 0 stampa + */
    if (a[i-1] )
        printf(" + ");
    }

/* polinomio nullo */
if (primo) printf("0");

printf("\n");
}

```

21

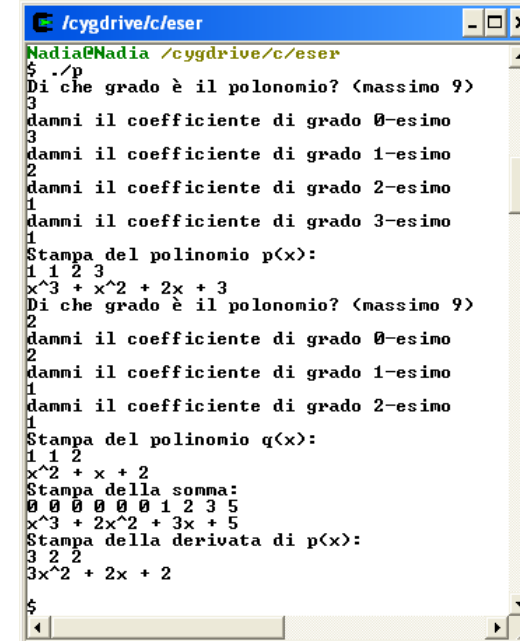
Parte 2. Esercizio 6



- Scrivere una funzione che determina quanti cambi di pendenza ci sono in un vettore t di interi di capacità N.
- Un cambio di pendenza si determina in un vettore quando vi è un elemento che è l'ultimo di una sequenza di elementi strettamente crescente e contemporaneamente il primo di una sequenza di elementi strettamente decrescente, oppure il contrario.
- Esempio
Input [1, 3, 7, 9, 6, 2, 4, 4, 5, 7, 3]
Output : 3
Ci sono tre cambi di pendenza in corrispondenza degli elementi : 9, 2 e 7.

23

Soluzione (5)

```

Cygdrive/c/eser
Nadia@Nadia /cygdrive/c/eser
$ ./p
Di che grado è il polinomio? <massimo 9>
3
dammi il coefficiente di grado 0-esimo
3
dammi il coefficiente di grado 1-esimo
2
dammi il coefficiente di grado 2-esimo
1
dammi il coefficiente di grado 3-esimo
1
Stampa del polinomio p(x):
1 1 2 3
x^3 + x^2 + 2x + 3
Di che grado è il polinomio? <massimo 9>
2
dammi il coefficiente di grado 0-esimo
2
dammi il coefficiente di grado 1-esimo
1
dammi il coefficiente di grado 2-esimo
1
Stampa del polinomio q(x):
1 1 2
x^2 + x + 2
Stampa della somma:
0 0 0 0 0 1 2 3 5
x^3 + 2x^2 + 3x + 5
Stampa della derivata di p(x):
3 2 2
3x^2 + 2x + 2
$

```

22

Soluzione (6)



Struttura del programma

- main
 - Lettura degli interi da tastiera ed inserimento in un array di N elementi (funzione `inputArray`)
 - Restituisce il numero di elementi effettivamente inseriti (riempimento dell'array)
 - `int inputArray(int a[], int cap)`
 - Calcolo del numero di cambi di pendenza mediante la funzione `calcola_cambi`
 - Esegue il calcolo sul numero di elementi effettivamente inseriti (riempimento)
 - `int calcola_cambi(int a[], int riemp);`
 - Stampa del risultato
 - parametro di ritorno della funzione `calcola_cambi`

24

Soluzione (6)



Algoritmo per la funzione `calcola_cambi`:

- Una variabile `count` per contare il numero di cambi di pendenza
- Una variabile intera `pend` indicata la pendenza corrente (crescente o decrescente)
- Si stabilisce una convenzione
 - `pend == 0` => crescente, `pend == 1` => decrescente
- Per `i` da 0 a `riemp-1`
 - Si confrontano le coppie di elementi adiacenti (`a[i]` `a[i+1]`)
 - Se `a[i+1]>a[i]` e la pendenza corrente è decrescente:
 - `count++`
 - `pend = 0`
 - Altrimenti non si fa nulla
 - Se `a[i+1]<a[i]` e la pendenza corrente è crescente:
 - `count++`
 - `pend = 1`
 - Altrimenti non si fa nulla

25

Soluzione (6)



- Occorre determinare la pendenza iniziale
- E' sufficiente confrontare la prima coppia?
- No, i primi due (o più) elementi potrebbero essere uguali!
- Variabile booleana `inizio` che vale `TRUE` per indicare che ancora non è stata stabilita la prima pendenza
 - Resettata quando viene trovata la prima coppia di elementi diversi (solo allora `pend` assume un valore significativo)

[1, 1, 8, 14, 12, 13, 54]

↑ ↑ `inizio = 1` `pend = x` `count = 0`

[1, 1, 8, 14, 12, 13, 54]

↑ ↑ `inizio = 0` `pend = 0` `count = 0`

[1, 1, 8, 14, 12, 13, 54]

↑ ↑ `inizio = 0` `pend = 0` `count = 0`

[1, 1, 8, 14, 12, 13, 54]

↑ ↑ `inizio = 0` `pend = 1` `count = 1`

26

Soluzione (6)



```
#include<stdio.h>
#define N 5
int inputArray(int a[], int c);
int calcola_cambi(int a[], int r);

main() {
    int a[N],r,p;
    r = inputArray(a,N);
    p = calcola_cambi(a,r);
    printf("Numero di cambi di pendenza = %d\n",p);
}

int calcola_cambi(int a[], int r){
    int i, cambi, pend, inizio;
    cambi = 0;
    inizio = 1;
    pend = 0; /* 0 == crescente, 1 == decrescente */
    printf("\n");
    for (i=0; i<r-1; i++) {
        printf("i: %d a[%d]: %d\n",i,i, a[i]);
```

27

Soluzione (6)



```
if (a[i+1]>a[i]){ /* se uguali la pendenza non cambia */
    printf("%d > %d\n",a[i+1], a[i]);
    if (inizio) {
        pend = 0;
        inizio = 0;
    }
    if (pend) {
        cambi++;
        pend = 0;
    }
} else if (a[i+1]<a[i]){
    printf("%d < %d\n",a[i+1], a[i]);
    if (inizio) {
        pend = 1;
        inizio = 0;
    }
    if (!pend){
        cambi++;
        pend = 1;
    }
}
}
return cambi; }
```

28

Soluzione (6)



```

C:\cygdrive\c\eser
Nadia@Nadia /cygdrive/c/eser
$ gcc pendenza.c -o pendenza

Nadia@Nadia /cygdrive/c/eser
$ ./pendenza
Quanti elementi vuoi inserire? <massi
5
dammi l'elemento 0-esimo
1
dammi l'elemento 1-esimo
2
dammi l'elemento 2-esimo
1
dammi l'elemento 3-esimo
3
dammi l'elemento 4-esimo
4

i: 0 a[0]: 1
2 > 1
i: 1 a[1]: 2
1 < 2
i: 2 a[2]: 1
3 > 1
i: 3 a[3]: 3
4 > 3
Numero di cambi di pendenza = 2
$
Nadia@Nadia /cygdrive/c/eser
$
    
```

29

Parte 3. Esercizio 9



- Si definisca una funzione che rimuove i duplicati in un array di interi ordinato. La funzione deve restituire il numero di elementi diversi contenuti nell'array.
- Compattare man mano gli elementi dell'array, utilizzando degli zero nelle ultime posizioni (vedere l'esempio mostrato a lezione).

Esempio

Input [1, 3, 3, 9, 16, 24, 25, 25, 33]

riempimento = 9

Output [1, 3, 9, 16, 24, 25, 33, 0, 0]

riempimento = 7

30

Soluzione (9)



Struttura del programma

- Main
 - Lettura degli interi da tastiera ed inserimento in un array di N elementi (funzione `inputArray`)
 - Restituisce il numero di elementi effettivamente inseriti (riempimento dell'array)

```
int inputArray(int a[], int cap)
```
 - Rimuove i duplicati mediante la funzione `rimuovi_duplicati`
 - Restituisce il numero di elementi diversi

```
int rimuovi_duplicati(int a[], int riemp);
```
 - Stampa del numero di elementi diversi
 - Stampa dell'array mediante la funzione `stampa`

```
void stampa(int a[], int riemp)
```

31

Soluzione (9)



[1, 1, 8, 14, 14, 23, 54]

↑↑ i=0 riemp = 7 a[0]==a[1]? SI

[1, 8, 14, 14, 23, 54, 0]

↑↑ i=1 riemp = 6 a[1]==a[2]? NO

[1, 8, 14, 14, 23, 54, 0]

↑↑ i=2 riemp = 6 a[2]==a[3]? SI

[1, 8, 14, 23, 54, 0, 0]

↑↑ i=3 riemp = 5 a[3]==a[4]? NO

32



- Algoritmo per la funzione `rimuovi_duplicati`
- Confronta coppie di elementi adiacenti
- Ogni volta che trova una coppia uguale, compatta gli elementi e pone zero nella posizione finale liberata
- In particolare
 - Per ogni coppia di elementi uguali ($a[i]=a[i+1]$):
 - Spostare tutti gli elementi successivi a $i+1$ di una posizione verso sinistra ($a[j]=a[j+1]$)
 - Aggiungere alla fine uno zero
 - Occorre ricordare fino a dove non ci sono zero
 - Se `num` rappresenta inizialmente il riempimento dell'array => per ogni coppia di elementi uguali `num--`
 - Ripetere il procedimento ripartendo dallo stesso indice i (in cui ora si troverà l'elemento $i+1$)
 - Potrebbero esserci elementi presenti anche più di due volte
 - Ripetere per i che va da 0 a `num-1`



```
#include<stdio.h>
#define N 10
int inputArray(int a[], int c);
int rimuovi_duplicati(int a[], int r);
void stampa (int a[], int r);

main() {
    int a[N],r, r1;
    r = inputArray(a,N);
    r1= rimuovi_duplicati(a,r);
    printf("Elementi diversi: %d\n",r1);
    printf("Risultato finale");
    stampa(a,r);
}
```



```
int rimuovi_duplicati(int a[], int r){
    int i, j;
    i=0;
    while (i<r-1) {
        if(a[i]==a[i+1]) {
            /* compatta l'array */
            for(j=i+1; j<r; j++)
                a[j] = a[j+1];
            a[r-1]= 0;
            r--;
            /* debug */
            stampa(a,r);
        } else i++;
    }
    return r;
}
```

```
icydrive/c/eser
Nadia@Nadia /cygdrive/c/eser
$ ./dup
Quanti elementi vuoi inserire? <nassi
5
dammi l'elemento 0-esimo
1
dammi l'elemento 1-esimo
2
dammi l'elemento 2-esimo
3
dammi l'elemento 3-esimo
3
dammi l'elemento 4-esimo
3
1      2      3      3
1      2      3
Elementi diversi: 3
Risultato finale: 1      2      3
Nadia@Nadia /cygdrive/c/eser
$
```