3 Array numerici

Questa dispensa propone esercizi sulla scrittura di algoritmi, in linguaggio C, utili alla comprensione dei vettori (*ingl.*, array). La dichiarazione di un vettore di elementi omogenei in C avviene grazie alla seguente dichiarazione:

```
void main(){
tipo_variabile nome_variabile[numero_elementi];

istruzioni;
}
```

dove `tipo_variabile` è il tipo degli elementi del vettore e `numero_elementi` è la lunghezza del vettore (considerata una costante da qui in poi). Solitamente la lunghezza del vettore viene dichiarato come una costante tramite la parola chiave `#define` per migliorare la leggibilità del codice.

Per accedere a singoli elementi dell'array utilizzeremo degli indici che varieranno tra 0 e `numero_elementi` - 1. Per accedere ad un elemento del vettore si utilizzano le parentesi quadre, ad esempio:

```
void main(){
int vettore[10];

vettore[1] = 4;
}
```

il precedente programma dichiara un vettore di interi di dimensione 10 e successivamente inserisce il valore 4 nel secondo elemento del vettore.

Un vettore ha dimensioni fisse. Esse devono essere specificate alla dichiarazione del programma. Non è possibile modificare le dimensioni di un vettore durante l'esecuzione del programma.

3.1 Esercizi

Esercizio 3.1

Scrivere un programma che, dati due vettori di interi di dimensione 10, ne costruisca un terzo di dimensione 20 i cui elementi di posizione pari siano gli elementi del primo vettore e gli elementi di posizione dispari siano gli elementi del secondo vettore.

Provare ad utilizzare un solo vettore di dimensione 20 per risolvere l'esercizio.

Esercizio 3.2

Scrivere un programma che acquisisca un certo numero di valori interi e positivi stabilito dall'utente (massimo 10), e poi stampi a video un vettore contenente tutti i numeri pari della sequenza e un vettore contente tutti i numeri dispari.

Esercizio 3.3

Scrivere un programma che acquisisca un numero stabilito dall'utente di voti di esami e si assicuri che tutti i voti inseriti siano nell'intervallo $[18, 30]$. Il numero massimo di voti inseribili è 20, ma l'utente potrebbe decidere di inserire meno voti. Il programma dovrà poi stampare a video un sommario simile a quello di seguito. La media troncata è la media calcolata non contando gli estremi (voti minimo e massimo).

```
STATISTICHE VOTI:

Numero esami sostenuti: ...
Media: ...
Media troncata: ...
Varianza: ...
Deviazione standard: ...
```

Per il calcolo la varianza, $\hat{\sigma}^2$, utilizzare la seguente formula:

$$\hat{\sigma}^2 = \frac{\sum_{i=1}^{N}(v_i - \hat{v})^2}{N-1} \tag{3.1}$$

dove v_i è l'i-esimo voto e $\hat{v}$ è la media campionaria dei voti. La deviazione standard è semplicemente la radice quadrata della varianza.

Esercizio 3.4

Scrivere un programma che, acquisita una sequenza di al massimo 50 numeri interi tra 1 e 100, stampi a video l'istogramma dei divisori. L'istogramma deve avere tutti i valori da 2 al valore massimo immesso diviso per 2. Il programma deve considerare solo i divisori propri, dove un divisore positivo di n diverso da n stesso è chiamato divisore proprio.

Un esempio dell'output del programma è:

```
Quanti numeri vuoi considerare: 5

Inserire il 1o numero: 6
Inserire il 2o numero: 4
Inserire il 3o numero: 3
Inserire il 4o numero: 9
Inserire il 5o numero: 12
2               | ***
3               | ***
4               | *
5               |
6               | *
```

Esercizio 3.5

Scrivere un programma che chieda all'utente di inserire un numero variabile (al massimo 10) di valori reali. Il programma dovrà assicurarsi che l'utente non possa inserire più di 10 valori.

Il programma dovrà cercare, durante l'acquisizione, la posizione del minimo e del massimo.

Infine, il programma dovrà stampare un sommario dei valori inseriti, con l'indicazione della posizione del minimo (indicata con un segno $-$) e del massimo (indicata con un segno $+$), il tutto formattato come indicato di seguito.

```
Quanti elementi vuoi inserire (max: 10)? 100
Il valore inserito è troppo grande!
Quanti elementi vuoi inserire (max: 10)? 4
Inserire il 1o valore: 100
Inserire il 2o valore: 20
Inserire il 3o valore: -1
Inserire il 4o valore: 5.3
  +| 100.00
   | 20.00
-  | -1.00
   | 5.30
```

Esercizio 3.6

Scrivere un programma che acquisisca un certo numero di valori stabilito dall'utente (massimo 10), e poi acquisisca un eguale quantità di indici, che specificano l'ordine in cui il programma dovrà stampare i valori acquisiti in precedenza. Per esempio:

```
Quanti elementi vuoi inserire (max: 10)? 5
Inserire il 1o valore: 10
Inserire il 2o valore: 20
Inserire il 3o valore: 30
Inserire il 4o valore: 40
Inserire il 5o valore: 50
Qual è il 1o valore che vuoi stampare? 3
Qual è il 2o valore che vuoi stampare? 2
Qual è il 3o valore che vuoi stampare? 1
```

```
Qual è il 4o valore che vuoi stampare? 4
Qual è il 5o valore che vuoi stampare? 5
30
20
10
40
50
```

Esercizio 3.7

Scrivere un programma che acquisisce un numero variabile di interi, con un massimo di 100. Il programma dovrà costruire un insieme (ossia controllare che non ci siano elementi ripetuti) a partire dagli elementi inseriti.

Esercizio 3.8

Implementare le operazioni di intersezione, unione, e differenza insiemistica tra due array acquisiti da tastiera, supponendo siano insiemi (ossia non abbiano elementi ripetuti).

Soluzioni

Soluzione dell'esercizio 3.1

```c
#include <stdio.h>

#define N 10

void main(){

    int v1[N], v2[N], v3[2*N];
    int i;

    //acquisizione dei due vettori in ingresso
    for(i = 0; i < N; i++){
        printf("Inserire il %d valore del primo vettore: ", i+1);
        scanf("%d", &v1[i]);
    }
    for(i = 0; i < N; i++){
        printf("Inserire il %d valore del primo vettore: ", i+1);
        scanf("%d", &v2[i]);
    }

    //ciclo di inserimento nel vettore in uscita
    for(i = 0; i < N; i++){
        v3[2 * i] = v1[i];
        v3[2 * i + 1] = v2[i];
    }

    //stampa del vettore rislutante
    printf("Vettore risultante: [ ");
    for(i = 0; i < 2 * N; i++){
        printf("%d ", v3[i]);
    }
    printf("]\n");

}
```

Soluzione dell'esercizio 3.2

```c
#include <stdio.h>

#define MAX_LEN 10

void main(){

    int pari[MAX_LEN];
    int dispari[MAX_LEN];
    int n_val;
    int i, valore;
    int n_pari = 0, n_disp = 0;

    //chiedo all'utente quanti valori intende immettere
    do{
        printf("Inserire il numero di valori da considerare: ");
        scanf("%d", &n_val);
    }while(n_val < 1 || n_val > MAX_LEN);
```

```
    //ciclo di acquisizione dei valori
    for (i = 0; i < n_val; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%d", &valore);
        //controllo se pari o dispari
        if (valore % 2 == 0) {
            pari[n_pari] = valore;
            n_pari++;
        }
        else {
            dispari[n_disp] = valore;
            n_disp++;
        }
    }

    //stampa dell'array pari
    printf("Numeri pari: ");
    for (i = 0; i < n_pari; i++) {
        printf("%d ",pari[i]);
    }
    printf("\n");

    //stampa dell'array dispari
    printf("Numeri dispari: ");
    for (i = 0; i < n_disp; i++) {
        printf("%d ",dispari[i]);
    }
    printf("\n");

}
```

Soluzione dell'esercizio 3.3

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

#define MAX_LEN 20

void main(){

    int n;
    int voti[MAX_LEN];
    int i;
    float sum;
    float media, media_troncata;
    int voto_minimo = 33;
    int voto_massimo = 10;
    float dev, varianza, deviazione_standard;

    //chiedo all'utente quanti esami vuole inserire
    do {
        printf("Inserire il numero degli esami: ");
        scanf("%d",&n);
    } while ( n < 3 || n > 20);

    // ciclo di acquiszione dei voti
    sum = 0;
    for (i = 0; i < n; i++) {
        //acquisisco un voto valido
```

```
        do {
            printf("Inserire il voto %d: ",i+1);
            scanf("%d",&voti[i]);
        } while ( voti[i] < 18 || voti[i] > 30);
        //aggiorno voto minimo e voto massimo
        if (voti[i] > voto_massimo)
            voto_massimo = voti[i];
        if (voti[i] < voto_minimo)
            voto_minimo = voti[i];
        //aggiorno la somma dei voti
        sum += voti[i];
    }

    //calcolo media e media troncata
    media = sum / n;
    media_troncata = (sum - voto_massimo - voto_minimo) / (n-2);

    //calcolo varianza e deviazione standard
    dev = 0;
    //ciclo per calcolo della sommatoria
    for (i = 0; i < n; i++)
        dev += (voti[i] - media) * (voti[i] - media);
    varianza = dev / (n - 1);
    deviazione_standard = sqrt(varianza);

    //stampa delle statistiche dei voti
    printf("\nSTATISTICHE VOTI\n");
    printf("Il numero degli esami: %d\n",n);
    printf("La media e': %f\n",media);
    printf("La media troncata e': %f\n",media_troncata);
    printf("La varianza e': %f\n",varianza);
    printf("La deviazione standard e': %f\n",deviazione_standard);

}
```

Soluzione dell'esercizio 3.4

```
#include <stdio.h>

#define LEN_DIVI 50
#define N_MAX 50

void main(){

    int numeri[N_MAX];
    int divisori[LEN_DIVI];
    int maxi = -1;
    int n_val = -1;
    int i,j;

    //chiedo all'utente quanti valori intende immettere
    while (n_val < 1 || n_val > N_MAX){
        printf("Inserire il numero di valori da considerare: ");
        scanf("%d",&n_val);
    }

    // Inizializzazione divisori
    for (i = 0; i < LEN_DIVI; i++)
        divisori[i] = 0;
```

```
    for (i = 0; i < n_val; i++) {
        //acquisizione di un valore
        do {
            printf("Inserire il %do valore: ",i+1);
            scanf("%d",&numeri[i]);
        } while (numeri[i] <= 0);
        //aggiornamento del massimo
        if (numeri[i] > maxi)
            maxi = numeri[i];
        //calcolo istogramma
        for (j = numeri[i] / 2; j > 1; j--)
            if (numeri[i] % j == 0)
                divisori[j-1]++;
    }

    //stampa istogramma
    for (i = 1; i < maxi / 2; i++) {
        printf("%d\t| ",i+1);
        for (j = 0; j < divisori[i]; j++)
            printf("*");
        printf("\n");
    }

}
```

Soluzione dell'esercizio 3.5

```
#include <stdio.h>

#define DIM 10

void main(){

    //dichiarazione variabili
    int i;
    int dim;
    int min_idx;
    int max_idx;
    float valori[DIM];

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);
        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    //acquisizione valori con ricerca di min max
    min_idx = 0; //assumiamo che min sia il primo elemento
    max_idx = 0; //assumiamo che max sia il primo elemento
    for (i = 0; i < dim; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%f", &valori[i]);

        if (valori[i] < valori[min_idx])
            min_idx = i;

        if (valori[i] > valori[max_idx])
            max_idx = i;
```

```
    }

    //stampa dei valori con indicazione del massimo e del minimo
    for (i = 0; i < dim; i++) {
        if (i == min_idx && i == max_idx)
            printf("- +|");
        else {
            if (i == min_idx)
                printf("-  |");

            if (i == max_idx)
                printf("  +|");
        }

        if (i != min_idx && i != max_idx)
            printf("   |");

        printf(" %.2f\n", valori[i]);
    }
}
```

Soluzione dell'esercizio 3.6

```
#include <stdio.h>

#define DIM 10

void main(){

    int i;
    int j; //indice usato solo per chiarezza, ma non necessario
    int indice[DIM];
    int valori[DIM];
    int dim;

    //acquisizione della dimensione
    do {
        printf("Quanti elementi vuoi inserire (max: %d)? ", DIM);
        scanf("%d", &dim);
        if (dim > DIM)
            printf("Il valore inserito e' troppo grande!\n");
    } while (dim > DIM);

    //acquisizione valori
    for (i = 0; i < dim; i++) {
        printf("Inserire il %do valore: ", i+1);
        scanf("%d", &valori[i]);
    }

    //acquisizione indice
    for (i = 0; i < dim; i++) {
        printf("Qual e' il %do valore che vuoi stampare? ", i+1);
        //tutti i valori dell'indice devono puntare ad elementi validi
        // quindi all'interno dei limiti 0,dim
        do {
            scanf("%d", &indice[i]);
            if (indice[i] > dim || indice[i] < 0)
                printf("Il valore inserito non e' in [0, %d]", dim);
        } while (indice[i] > dim || indice[i] < 0);
```

```
        //si assume che l'utente che vuole indicare il primo elemento (ad
        // esempio), inserisca il numero 1, quindi dobbiamo decrementare
        // tutto di 1: 1o, 2o, 3o... -> 0, 1, 2
        indice[i] = indice[i] - 1;
    }

    //stampa in ordine stabilito dall'indice
    for (i = 0; i < dim; i++)
    {
        j = indice[i];
        printf("%d\n", valori[j]);
    }
}
```

Soluzione dell'esercizio 3.7

```
#include <stdio.h>

#define MAX_LEN 100

void main(){

    int n;
    int lista[MAX_LEN];
    int insieme[MAX_LEN];
    int lungh_ins;
    int i,j;
    int trovato;

    //chiedo all'utente quanti elementi vuole inserire
    do {
        printf("Inserire il numero degli elementi da inserire: ");
        scanf("%d", &n);
    } while ( n < 0 || n > 100);

    //ciclo di acquisizione degli elementi
    for (i = 0; i < n; i++) {
            printf("Inserire il %do numero : ", i+1);
            scanf("%d", &lista[i]);
    }

    //scorro gli elementi in lista ed inserisco in insieme
    lungh_ins = 1;
    insieme[0] = lista[0];
    for (i = 1; i < n; i++) {
        //controllo se l'elemento  gi in insieme
        trovato = 0;
        for (j = 0; j < lungh_ins; j++)
            if (lista[i] == insieme[j])
                trovato = 1;
        //se non  ripetuto insrisco
        if (!trovato) {
            insieme[lungh_ins] = lista[i];
            lungh_ins++;
        }
    }

    //stampa degli elementi dell'insieme
    printf("[ ");
    for (i = 0; i < lungh_ins;i++)
```

```
        printf("%d ",insieme[i]);
    printf("]\n");

}
```

Soluzione dell'esercizio 3.8

```
#define DIM 30

#include <stdio.h>

void main(){

    // indici per scansione
    int i, j;

    // dimensione effettiva degli array
    int dim_a, dim_b;

    // flag utile per le ricerche di elementi
    int trovato;

    // array e insiemi A e B
    int lista[DIM];
    int A[DIM];
    int B[DIM];

    // dimensione effettiva degli insiemi
    int len_a = 0;
    int len_b = 0;
    int len_u = 0; //dimensione unione
    int len_i = 0; //dimensione intersezione
    int len_d = 0; //dimensione differenza

    // altri insiemi
    int unione[2*DIM];    // A u B
    int intersezione[DIM];// A ^ B
    int differenza[DIM];  // B \ A

    /*
     * ACQUISIZIONE DELLA DIMENSIONE DELLA PRIMA LISTA
     */
    do {
        printf("Quanti elementi vuoi inserire nella prima lista (max: %d)? ",
               DIM);
        scanf("%d", &dim_a);

        if (dim_a > DIM)
            printf("Il valore inserito  troppo grande!\n");
    } while (dim_a > DIM);

    /*
     * ACQUISIZIONE DELLA PRIMA LISTA DI ELEMENTI
     */
    for (i = 0; i < dim_a; i++) {
        printf("Inserire il %do elemento: ", i+1);
        scanf("%d", &lista[i]);
    }

    /*
```

```
 * CONVERSIONE IN INSIEME A
 */
for (i = 0; i < dim_a; i++) {
    trovato = 0;

    //ricerco il valore i-esimo nella lista
    for (j = 0; !trovato && j < len_a; j++)
        trovato = (lista[i] == A[j]);

    //se non trovato, lo inserisco e incremento l'indice
    if (!trovato) {
        //gi che ci sono, lo inserisco anche nell'unione
        unione[len_a] = lista[i];

        A[len_a] = lista[i];
        len_a++;
    }
}

/*
 * A QUESTO PUNTO A COINCIDE CON L'INSIEME UNIONE, AL QUALE DOVR
 * AGGIUNGERE GLI ELEMENTI DI B SENZA RIPETIZIONI
 */
len_u = len_a;

// stampa l'insieme A
printf ("Insieme A = {");
for (i = 0; i < len_a; i++)
    printf("%d ", A[i]);
printf ("}\n");

/*
 * ACQUISIZIONE DELLA DIMENSIONE DELLA SECONDA LISTA
 */
do {
    printf("Quanti elementi vuoi inserire nella seconda lista (max: %d)? ",
           DIM);
    scanf("%d", &dim_b);

    if (dim_b > DIM)
        printf("Il valore inserito  troppo grande!\n");
} while (dim_b > DIM);

/*
 * ACQUISIZIONE DELLA SECONDA LISTA DI ELEMENTI
 */
for (i = 0; i < dim_b; i++) {
    printf("Inserire il %do elemento: ", i+1);
    scanf("%d", &lista[i]);
}

/*
 * CONVERSIONE IN INSIEME B E CALCOLO UNIONE, INTERSEZIONE,
 * DIFFERENZA
 */
for (i = 0; i < dim_b; i++) {
    trovato = 0;

    //ricerco il valore i-esimo nella lista
    for (j = 0; !trovato && j < len_b; j++)
        trovato = (lista[i] == B[j]);
```

```
        //se non trovato, lo inserisco e incremento l'indice
        if (!trovato) {
            B[len_b] = lista[i];
            len_b++;
        }

        /*
         * UNIONE: Tutti gli elementi in A e tutti gli elementi in B
         * senza ripetizioni.
         */
        trovato = 0;

        //ricerco il valore i-esimo nella lista
        for (j = 0; !trovato && j < len_u; j++)
            trovato = (unione[j] == lista[i]);

        //se non trovato, allora va aggiunto all'unione
        if (!trovato) {
            unione[len_u] = lista[i];
            len_u++;
        }

        /*
         * INTERSEZIONE: Tutti gli elementi che sono sia in A che in
         * B, senza ripetizioni.
         */
        trovato = 0;

        //uso len_b-1 perch len_b  gi stato incrementato
        for (j = 0; !trovato && j < len_a; j++)
            trovato = (A[j] == B[len_b-1]);

        //trovato anche in A: allora fa parte dell'intersezione
        if (trovato) {
            trovato = 0;

            //controllo per non inserire duplicati in intersezione
            for (j = 0; !trovato && j < len_i; j++)
                trovato = (intersezione[j] == B[len_b-1]);

            if (!trovato) {
                intersezione[len_i] = B[len_b-1];
                len_i++;
            }
        } else { //non trovato in A: allora fa parte della differenza
            /*
             * DIFFERENZA: Tutti gli elementi che sono in B, tranne
             * quelli che sono in A.
             */
            trovato = 0;

            //controllo per non inserire duplicati in differenza
            for (j = 0; !trovato && j < len_d; j++)
                trovato = (differenza[j] == B[len_b-1]);

            if (!trovato) {
                differenza[len_d] = B[len_b-1];
                len_d++;
            }
        }
    }
```

```
    // stampa l'insieme B
    printf ("Insieme B = { ");
    for (i = 0; i < len_b; i++)
        printf("%d ", B[i]);
    printf ("}\n");

    // stampa l'insieme unione
    printf ("unione = { ");
    for (i = 0; i < len_u; i++)
        printf("%d ", unione[i]);
    printf ("}\n");

    // stampa l'insieme intersezione
    printf ("intersezione = { ");
    for (i = 0; i < len_i; i++)
        printf("%d ", intersezione[i]);
    printf ("}\n");

    // stampa l'insieme differenza
    printf ("differenza = { ");
    for (i = 0; i < len_d; i++)
        printf("%d ", differenza[i]);
    printf ("}\n");
}
```