INFORMATICA – Preparazione alla prova di teoria

ESERCIZIO 1

La struttura `struct punto { float x, y; }` descrive punti in un piano cartesiano, mentre la struttura `struct triangolo { punto v1, v2, v3; }` descrive triangoli rappresentandone i tre vertici nel piano. Scrivete una funzione che ha come parametro formale una variabile di tipo `triangolo` e restituisce `true` se il triangolo passato per argomento è *equilatero*, `false` altrimenti.

SOLUZIONE ESERCIZIO 1

```
//calcola la distanza tra due punti
float distanza( punto a, punto b ) {
    return sqrt( pow( a.x - b.x, 2 ) - pow( a.y - b.y, 2 ) );
}

//valuta se i tre lati del triangolo t hanno la stessa lunghezza
bool equilatero( triangolo t ) {
    return( distanza( t.v1, t.v2 ) == distanza( t.v2, t.v3 ) &&
            distanza( t.v2, t.v3 ) == distanza( t.v1, t.v3 ) );
}
```

ESERCIZIO 2

Usando la struct `triangolo` e la funzione richiesta all'esercizio precedente, scrivete un frammento di codice che legge dall'utente 100 triangoli (inseriti specificando per ogni triangolo le coordinate dei vertici) e li memorizza in un array. Successivamente, il programma deve spostare in coda all'array i soli triangoli *equilateri*.

SOLUZIONE ESERCIZIO 2

```
triangolo T[100];

//leggiamo uno per uno i vertici dei triangoli dell'array
for ( int i = 0; i < 100; i++ ) {
    cout << "Coordinate del primo vertice:\t";
    cin >> (T[i].v1).x >> (T[i].v1).y;
    cout << "Coordinate del secondo vertice:\t";
    cin >> (T[i].v2).x >> (T[i].v2).y;
    cout << "Coordinate del terzo vertice:\t";
    cin >> (T[i].v3).x >> (T[i].v3).y;
}
```

continua alla pagina successiva ⇒

```
for ( int i = 0, j = 99; i < j; i++ ) {            //sposta sul fondo i triangoli equilateri
    if ( equilatero( T[i] ) ) {
        swap( T[j], T[i] );
        i--;
        j--;
    }
}

void swap ( triangolo& p, triangolo& q ) {   //scambia due triangoli
    triangolo r = p;
    p = q;
    q = r;
}
```

———— ESERCIZIO 3 ————

Scrivete un frammento di codice che legge dall'utente una sequenza di interi inseriti uno per riga e terminata da 0. Dopo l'acquisizione di tale sequenza, il programma deve scrivere "OK" se la sequenza conteneva *quattro o più interi negativi inseriti consecutivamente*, "KO" altrimenti.

SOLUZIONE ESERCIZIO 3

```
int k = 0, n;
do {
    cin >> n;
    if ( k < 4 )
        k = ( n < 0 ? k + 1 : 0 );
} while ( n != 0 );
cout << ( k == 4 ? "OK" : "KO" ) << endl;
```

———— ESERCIZIO 4 ————

Dati due array di interi `A` e `B` entrambi di dimensione `d`, il loro *shuffle* è un *array* di interi `C` di dimensione $2 \cdot$ `d` in cui `C[0]` contiene `A[0]`, `C[1]` contiene `B[0]`, `C[2]` contiene `A[1]`, `C[3]` contiene `B[1]`, `C[4]` contiene `A[2]`, `C[5]` contiene `B[2]` e così via. Scrivete la funzione `int* shuffle( int* A, int* B, int d )` che restituisce lo shuffle dei due array passati per argomento.

SOLUZIONE ESERCIZIO 4

```
int* shuffle( int* A, int* B, int d ) {
    int* C = new int[2*d];
    for( int i = 0; i < d; i++ ) {
        C[2*i] = A[i];
        C[2*i + 1] = B[i];
    }
    return C;
 }
```

———— ESERCIZIO 5 ————

Scrivete la funzione

`float funzione( float d, float i, int a )`

che, dato un capitale iniziale `d`, un interesse annuo `i` (espresso in percentuale) e un numero di anni `a`, restituisce il capitale iniziale accresciuto dagli interessi maturati dopo il numero di anni passati. Ad esempio, dopo un anno, l'importo restituito dalla funzione sarà

$$\mathtt{d}' = \mathtt{d} + \mathtt{d} * \mathtt{i}/100,$$

dopo il secondo anno sarà

$$\mathtt{d}'' = \mathtt{d}' + \mathtt{d}' * \mathtt{i}/100,$$

e così via.

SOLUZIONE ESERCIZIO 5

```
//diamo una soluzione ricorsiva
float funzione( float d, float i, int a ) {
    if ( a == 0 )
        return d;
    else
        return funzione( d + d * i/100, i, a - 1 );
}
```

———— ESERCIZIO 6 ————

Data la successione di interi $X = x_1, \ldots, x_n$, la sua *somma di Cesàro* è $\text{Cesaro}(X) = \sum_{k=1}^{n} \left(\sum_{i=1}^{k} x_i/k \right)$. In parole povere, Cesàro(X) è la somma delle medie parziali della successione X. Scrivete la funzione

`float Cesaro( int* X, int d )`

che restituisce la somma di Cesàro della successione di interi contenuta nell'array `X` di dimensione `d`.

SOLUZIONE ESERCIZIO 6

```
float Cesaro( int* X, int d ) {
    float ces = 0;
    for ( int k = 1; k <= d; k++ ) {        //attenzione agli estremi dei for
        ces = ces + mean( X, k );
    }
    return ces;
}

//funzione per il calcolo della media
float mean( int* X, int n ) {
    float sum = 0;
    for ( int i = 0; i < n; i++ )
            sum = sum + X[i];
    return sum/n;
}
```

ESERCIZIO 7

La sequenza $\{f_n\}_{n\geq 0}$ dei numeri di Fibonacci è definita da:

$$f_n = \begin{cases} n & \text{se } n \leq 1 \\ f_{n-1} + f_{n-2} & \text{se } n > 2. \end{cases}$$

Scrivete la funzione `void Fib( int* X, int n )` che riempie da sinistra a destra l'array `X` di dimensione `n` coi primi `n` numeri di Fibonacci.

SOLUZIONE ESERCIZIO 7

```
//non usiamo una soluzione ricorsiva dato che abbiamo modo di immagazzinare i numeri calcolati
void Fib( int* X, int n ) {
    for ( int i = 0; i < n; i++ )
        X[i] = ( i <= 1 ? i : X[i - 1] + X[i - 2] );
}
```