Pseudocodice;

La miglior soluzione per **leggere e capire** facilmente gli esercizi di programmazione delle selezioni scolastiche.

Lo Staff delle OII

6 novembre 2018

Come anticipato nell'ultima revisione del regolamento, a partire dal 15 novembre 2018 la fase scolastica delle Olimpiadi Italiane di Informatica si rinnova.

La Selezione Scolastica è da anni suddivisa in tre parti: esercizi di carattere logico matematico, esercizi di carattere algoritmico ed esercizi di programmazione. Da quest'anno, gli **esercizi di programmazione** non saranno più nei linguaggi C/C++ e Pascal, bensì verranno presentati con il formalismo (o meglio l'*informalismo*) dello **pseudocodice**.

Con *pseudocodice* intendiamo un linguaggio che ci permette di descrivere programmi usando una sintassi naturale, umana, senza le rigide regole di un linguaggio di programmazione.

Attenzione! Questo cambiamento si riferisce **solo agli esercizi di programmazione** contenuti nella **selezione scolastica**. La fase territoriale e nazionale, naturalmente, rimangono invariate e richiedono sempre la scrittura di programmi veri.

È importante notare che, sebbene le fasi successive alla scolastica facciano uso di linguaggi veri e propri, in questo caso esiste una fondamentale distinzione: ciò che valutiamo in questa gara infatti è l'abilità nel **leggere e capire** del codice già scritto, non quella di **scriverne**. Siamo convinti che lo pseudocodice sia particolarmente adatto allo scopo di questa gara.

Attenzione! Questo cambiamento **non renderà più difficile** svolgere gli esercizi di programmazione. Lo pseudocodice che useremo è infatti stato pensato ed introdotto nella selezione proprio con l'obiettivo di **rendere più accessibili** gli esercizi di programmazione, allargando l'insieme di studenti che sono in grado di risolverli.

Mantenendo gli esercizi nei linguaggi C/C++ e Pascal, gli studenti con gli strumenti necessari per svolgere questo tipo di esercizi sarebbero soltanto *coloro ai quali è stato insegnato uno di quei tre linguaggi*. Con lo pseudocodice, invece, questi esercizi diventano alla portata di un ben più ampio numero di studenti: *tutti quelli ai quali è stato insegnato un qualsiasi linguaggio di programmazione strutturato*, sia esso C, C++, Pascal, Java, Python, o uno dei tanti altri linguaggi che vengono quotidianamente insegnati nelle scuole italiane.

L'intera selezione scolastica viene resa **più equa** con questo cambiamento: prima, gli studenti affrontavano gli esercizi di *carattere logico matematico* e quelli di *carattere algoritmico* con gli stessi strumenti. Per quanto riguarda gli *esercizi di programmazione*, invece, gli studenti che avevano ricevuto una formazione in linguaggio C, C++ o Pascal risultavano **avvantaggiati** rispetto a quelli che avevano invece studiato altri linguaggi a scuola.

In ogni esercizio, il Comitato curerà lo pseudocodice in modo attento a mantenere estrema chiarezza e semplicità, evitando allo stesso tempo di introdurre ambiguità.

Esempi della sintassi utilizzata:

Pseudocodice	Significato
`a` ← `3;` `a` ← `a + 1;`	assegnamento del valore `3` ad `a`, incremento del valore di `a`
`leggi(a);` `scrivi(a);`	istruzioni per leggere dati da tastiera e scrivere dati sullo schermo
`scrivi("ciao");` `scrivi("ciao", var);`	scrive la stringa `ciao` a video scrive `ciao` e poi il contenuto della variabile `var`
`istruzione;`	il punto e virgola, per convenzione, indica la fine di un'istruzione
`array[7]` ← `0;` `mat[5][28]` ← `1;`	esempi di inizializzazione di un elemento di un array e di una cella di una matrice
`+ - * / div mod`	questi simboli indicano rispettivamente le operazioni aritmetiche di somma, differenza, prodotto, divisione, divisione intera (quoziente) e resto della divisione
$= < > \leq \geq \neq$	sono i simboli che indicano i sei tipi di confronto: uguale, minore, maggiore, minore o uguale, maggiore o uguale, diverso
(`a` ≤ 3) **oppure** (`a` ≥ 5) (`a` $\neq 0$) **e** (`b` $\neq 0$) **non** (`a`)	esempi di operazioni logiche: **non** per il NOT logico, **e** per l'AND, **oppure** per l'OR

Attenzione! Queste definizioni non sono un "contratto vincolante". Il Comitato Olimpico potrebbe ritenere utile usare notazioni speciali ai fini della chiarezza e facilità di lettura dell'esercizio. Per esempio:

(`a` ≤ 3) **oppure** (`b` è un numero primo)

Esempi di strutture di controllo (equivalenti di `if`, `while`, etc ...)

Pseudocodice	Significato
se `condizione` **allora** ... **fine se**	è un costrutto condizionale
finché `condizione` **ripeti** ... **fine finché**	è un ciclo a condizione iniziale (pre-posta)
ripeti ... **finché** `condizione`	è un ciclo a condizione finale (post-posta)
funzione `funzione(p1:tipo, ..., pn:tipo):tipo` ... `restituisci valore_restituito;` **fine funzione**	sintassi di una funzione
procedura `nome_procedura(p1:tipo, ..., pn:tipo)` ... `restituisci;` **fine procedura**	sintassi di una procedura (funzione che non restituisce un valore)

Attenzione! Anche per le strutture di controllo, il Comitato potrebbe adottare notazioni speciali qualora sia opportuno per spiegare l'algoritmo in modo più leggibile. L'importante, al solito, è che il codice non sia ambiguo o aperto ad interpretazioni. Per esempio, questo codice stampa i numeri `2`, `4`, `6`, `8`, `10`:

per ogni numero dispari `x` **da** `1` **a** `9` **ripeti**
    `scrivi(x + 1);`
fine per ogni

Esempi di pseudocodice adattati dalle Selezioni Scolastiche 2017

Esercizio N°6

Si consideri la seguente funzione:

C/C++

```c
int fun(int p) {
 printf("%d -> ", p);
 if (p%2 == 0)
   printf("condizione 1\n");
 if (p == 7)
   printf("condizione 2\n");
 else if ((p-5)%2 == 0)
   printf("condizione 3\n");
 return p;
}
```

Pascal

```pascal
function fun(p:integer):integer;
 begin
  write(p);
  write(' -> ');
  if (p mod 2 = 0) then
   writeln('condizione 1');
  if (p = 7) then
   writeln('condizione 2')
  else if ((p-5) mod 2 = 0) then
        writeln('condizione 3');
  fun:=p;
end;
```

La versione in pseudocodice segue:

Pseudocodice esercizio 6

```
1: funzione fun(p:intero):intero
2:     scrivi(p, " -> ");
3:     se p mod 2 = 0 allora
4:         scrivi("condizione 1");
5:     fine se
6:     se p = 7 allora
7:         scrivi("condizione 2");
8:     altrimenti
9:         se (p - 5) mod 2 = 0 allora
10:            scrivi("condizione 3");
11:        fine se
12:    fine se
13:    restituisci p;
14: fine funzione
```

Quale delle seguenti affermazioni è errata?

(a) La funzione, se p è pari, scrive a video il valore di p seguito dalla stringa `-> condizione 1` e restituisce p

(b) La funzione, se p non è dispari, scrive a video il valore di p seguito dalla stringa `-> condizione 2` e restituisce p

(c) La funzione, se p è 7, scrive a video il valore di p seguito dalla stringa `-> condizione 2` e restituisce p

(d) La funzione, se p è dispari, scrive a video p seguito dalla stringa `-> condizione 2` o `-> condizione 3` e restituisce p

Esercizio n°7

È dato il seguente programma:

C/C++

```c
#include <stdio.h>
#include <math.h>
int main()
{
int x,y,a,p;
float l, d;
x=20;
y=10;
a=x*y;
p=2*x+2*y;
l=p/4;
d=sqrt(2)*l;
printf("%f cm", d);
if(d*2 -720 == 0) d=2;
   else d=1;
return 0;
}
```

Pascal

```pascal
program E7(input,output);
var x,y,a,p:integer;
l,d:real;
begin
  x:=20;
  y:=10;
  a:=x*y;
  p:=2*x+2*y;
  l:=p/4;
  d:=sqrt(2)*l;
  writeln(d:7:6, ' cm');
  if( d * 2 -720 = 0) then d:=2
    else d:=1;
end.
```

La versione in pseudocodice segue:

Pseudocodice esercizio 7

```
1: x ← 20;
2: y ← 10;
3: a ← x * y;
4: p ← 2 * x + 2 * y;
5: l ← p / 4;
6: d ← sqrt(2) * l;
7: scrivi(d, " cm");
8: se (d * 2 - 720) = 0 allora
9:    d ← 2;
10: altrimenti
11:    d ← 1;
12: fine se
```

Cosa viene visualizzato a video dall'esecuzione del programma qui sopra?

(a) 2.000000 cm

(b) 3.000000 cm

(c) 21.213203 cm

(d) 36.243204 cm

Esercizio n°8

Si consideri la seguente funzione:

C/C++

```c
int myster(int c, int d) {
  if(c==d)
    return c;
  if(c>d)
    return myster(c-d, d);
  return myster(c, d-c);
}
int mcm(int a, int b) {
  return myster(b,a);
}
```

Pascal

```pascal
function myster(c:longint; d:longint):
  longint;
begin
  if c=d then myster:=c
  else
  if c>d then myster:=myster(c-d, d)
  else myster:=myster(c, d-c);
end;

function mcm(a:longint; b:longint):
  longint;
begin
 mcm:= myster(b,a);
end;
```

La versione in pseudocodice segue:

Pseudocodice esercizio 8

```
1: funzione myster(c:intero, d:intero):intero
2:     se c = d allora
3:         restituisci c;
4:     fine se
5:     se c > d allora
6:         restituisci myster(c - d, d);
7:     fine se
8:     restituisci myster(c, d - c);
9: fine funzione

10: funzione mcm(a:intero, b:intero):intero
11:     restituisci myster(b, a);
12: fine funzione
```

Quale delle seguenti modifiche fa sì che la funzione mcm ritorni il minimo comune multiplo tra a e b?

(a) sostituire `myster(b,a);` con `myster(a,b);`

(b) sostituire `myster(b,a);` con `(a*b)/myster(b,a);`

(c) sostituire `myster(b,a);` con `myster(a-b,b);`

(d) sostituire `myster(b,a);` con `myster(a,b-a);`