Pseudocodice 2.0;

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

Lo Staff delle OII

25 ottobre 2019

Da diversi anni, gli esercizi della Selezione Scolastica sono suddivisi in tre parti:

- Esercizi di carattere logico matematico
- Esercizi di carattere algoritmico
- Esercizi di programmazione

A partire dalla Selezione Scolastica di novembre 2018 abbiamo introdotto un importante cambiamento relativo alla forma in cui gli *esercizi di programmazione* venivano proposti: invece di fornire del codice vero e proprio (in due dei linguaggi più "popolari" tra quelli insegnati nelle classi italiane: **C** e **Pascal**) abbiamo cominciato a utilizzare uno **pseudocodice**.

Con pseudocodice intendiamo un linguaggio che 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.

Con lo pseudocodice, infatti, gli esercizi di programmazione diventano alla portata di *tutti gli studenti 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.

Novità 2019

A partire da quest'anno, grazie ai preziosi consigli e alle osservazioni fatte dai referenti territoriali alle ultime finali nazionali, il Comitato ha introdotto due ulteriori modifiche, di nuovo relative solo agli *esercizi di programmazione* della Selezione Scolastica.

Reintroduzione del codice "vero" come supporto allo pseudocodice

Il motivo di questa modifica è che, nonostante i migliori sforzi, c'è sempre una minuscola possibilità che lo pseudocodice lasci a spazio all'interpretazione: ad esempio il costrutto "**finché** `condizione` **ripeti**" ha fatto sorgere dei dubbi ad alcuni studenti in gara su come iterasse il ciclo: se con la condizione vera oppure con quella falsa. Vogliamo evitare qualsiasi altro dubbio che potrebbe sorgere in futuro.

Questa nostra modifica, sebbene possa sembrare un "ritorno alle origini", in realtà ci permette di avere tutti i vantaggi dello pseudocodice (ovvero di estendere il pool di studenti che vengono messi in condizione di provare a risolvere gli esercizi di programmazione) mantenendo comunque l'assoluta formalità e non ambiguità degli esercizi. Infatti, in gara, il codice sarà il punto di riferimento ed il principale strumento a disposizione dei referenti scolastici per risolvere qualsiasi dubbio degli studenti.

Il linguaggio scelto per svolgere questo compito è il **C**.

Aggiornamento del nostro pseudocodice

Come ci è stato fatto notare da una referente territoriale, anche alle Olimpiadi del Problem Solving (uno dei tanti progetti ministeriali "amici" delle Olimpiadi di Informatica) viene usato lo pseudocodice. Il Comitato ha quindi preso l'iniziativa di provare ad uniformare lo pseudocodice in uso alle OII con quello che già da anni viene usato alle OPS. Sperabilmente, questo faciliterà ulteriormente la partecipazione di studenti OPS alle OII, e viceversa.

Per via di limitazioni tecniche non è possibile uniformare completamente il nostro pseudocodice a quello delle OPS quindi rimangono alcune differenze, come il fatto che le nostre dichiarazioni di variabili saranno in linee multiple invece che in un'unica linea.

Per riferimento, è possibile consultare la guida allo pseudocodice in uso alle OPS al seguente indirizzo (da pagina 56):

https://www.olimpiadiproblemsolving.it/site/documenti/18-19/GUIDA_OPS_2019.pdf

Seguono degli esempi della sintassi utilizzata:

Pseudocodice	Significato
variable a: integer	dichiarazione della variabile a di tipo intero
variable w: integer[14]	dichiarazione della variabile w di tipo array di interi, fatto da 14 elementi
variable mat: char[10][10]	dichiarazione della variabile mat di tipo matrice di caratteri di dimensione 10×10
a $\leftarrow$ 3	assegnamento del valore 3 ad a,
a $\leftarrow$ a + 1	incremento del valore di a
output a	istruzione per scrivere dati sullo schermo
output ''ciao''	scrive la stringa ciao a video
output ''ciao'', var	scrive ciao e poi il contenuto della variabile var
array[7] $\leftarrow$ 0	esempi di inizializzazione di un elemento di un array
mat[5][28] $\leftarrow$ 1	e di una cella di una matrice
$+ \quad - \quad \times$ / MOD	questi simboli indicano rispettivamente le operazioni aritmetiche di somma, differenza, prodotto, divisione e resto della divisione intera
$= \quad < \quad > \quad \leq \quad \geq \quad \neq$	sono i simboli che indicano i sei tipi di confronto: uguale, minore, maggiore, minore o uguale, maggiore o uguale, diverso
(a $\leq$ 3) **or** (a $\geq$ 5)	esempi di operazioni logiche: ! per il NOT logico,
(a $\leq$ 3) **and** (a $\leq$ 5)	**and** per l'AND,
! (a)	**or** 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 $\leq$ 3) **or** (b è un numero primo)

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

Pseudocodice	Significato

if `condizione` **then**
...
end if
	è un costrutto condizionale

while `condizione` **do**
...
end while
	è un ciclo a condizione iniziale (pre-posta), ovvero il corpo viene eseguito ogni volta che `condizione` è VERA

repeat ... **until** `condizione`	è un ciclo a condizione finale (post-posta), ovvero il corpo viene eseguito fino a che `condizione` diventa FALSA
function `funzione(p1:tipo, ..., pn:tipo):tipo` ... **return** `valore_restituito` **end function**	sintassi di una funzione
procedure `nome_procedura(p1:tipo, ..., pn:tipo)` ... **return** **end procedure**	sintassi di una procedura (funzione che **non** restituisce un valore)

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: function FUN(p: integer) → integer
2:     output "p ->", p
3:     if p MOD 2 = 0 then
4:         output "condizione 1"
5:     end if
6:     if p = 7 then
7:         output "condizione 2"
8:     else
9:         if p − 5 MOD 2 = 0 then
10:            output "condizione 3"
11:        end if
12:    end if
13:    return p
14: end function
```

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: variable x: integer
2: variable y: integer
3: variable a: integer
4: variable p: integer
5: variable l: float
6: variable d: float
7: x ← 20
8: y ← 10
9: a ← x × y
10: p ← 2 × x + 2 × y
11: l ← p / 4
12: d ← SQRT(2) × l
13: output d, '' cm''
14: if d × 2 − 720 = 0 then
15:     d ← 2
16: else
17:     d ← 1
18: end if
```

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: function MYSTER(c: integer, d: intger) → integer
2:     if c = d then
3:         return c
4:     end if
5:     if c > d then
6:         return MYSTER(c − d, d)
7:     end if
8:     return MYSTER(c, d − c)
9: end function
10: function MCM(a: integer, b: integer) → integer
11:     return MYSTER(b, a)
12: end function
```

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);`

Esempi di pseudocodice adattati dalle Selezioni Scolastiche 2018

Esercizio n°6 Dato il seguente pseudocodice:

Pseudocodice esercizio 6

```
1: variable conta: integer
2: variable alfa: integer
3: variable beta: integer
4: conta ← 0
5: alfa ← 0
6: beta ← 0
7: while conta < 29 do
8:     if conta MOD 3 = 1 then
9:         alfa ← alfa + 2
10:    else
11:        beta ← beta − 1
12:    end if
13:    conta ← conta + 2
14: end while
```

calcolare il numero NUMRIP di volte per cui viene eseguito il ciclo **repeat-until**.

Esercizio n°7

Data la seguente funzione, dove N è la dimensione dell'array `a`:

Pseudocodice esercizio 7

```
1: function F(a: array of integer, N: integer) → integer
2:     variable palo: integer
3:     variable dado: integer
4:     variable i: integer
5:     palo ← −1
6:     dado ← −1
7:     i ← −1
8:     while i ≤ N − 1 do
9:         if a[i] MOD 2 = 0 then
10:            if a[i] > palo then
11:                palo ← a[i]
12:            else
13:                if a[i] > dado then
14:                    data ← a[i]
15:                end if
16:            end if
17:            i ← i + 1
18:        end if
19:    end while
20:    return palo + dado
21: end function
```

indicare quale tra le seguenti affermazioni è FALSA:

(a) La funzione f restituisce 23 se riceve in ingresso l'array 1, 2, 6, 10, 22

(b) La funzione f restituisce la massima somma di due elementi dell'array

(c) La funzione f restituisce numeri sia pari sia dispari

(d) La funzione f non può restituire valori inferiori a -2

Esercizio n°8

Si vuole scrivere una procedura che visualizzi a video una matrice 5×5 avente il contenuto seguente:

	1	1	1	
		1		
		1		

ma nella procedura qui riportata manca un pezzo.

Pseudocodice esercizio 8

```
1: procedure STAMPAT
2:     variable mat: integer[5][5]
3:     variable i: integer
4:     variable j: integer
5:     i ← 1
6:     while i ≤ 5 do
7:         j ← 1
8:         while j ≤ 5 do
9:             mat[i][j] ← 0
10:            j ← j + 1
11:        end while
12:        i ← i + 1
13:    end while
14:    i ← 1
15:    while i ≤ 5 do
16:        j ← 1
17:        while j ≤ 5 do
18:            . . . manca un pezzo . . .
19:            j ← j + 1
20:        end while
21:        i ← i + 1
22:    end while
23:    i ← 1
24:    while i ≤ 5 do
25:        j ← 1
26:        while j ≤ 5 do
27:            if mat[i][j] = 0 then
28:                output ‘‘_spazio_’’
29:            else
30:                output mat[i][j]
31:            end if
32:            j ← j + 1
33:        end while
34:        i ← i + 1
35:    end while
36: end procedure
```

Quale tra i seguenti pezzi è quello da inserire?

A)

```
1: if i − 1 ≠ 0 then
2:     if i − 1 = 1 then
3:         if j ≠ 0 and j ≠ 4 then
4:             mat[i − 1][j] ← 1
5:         end if
6:     end if
7:     if i − 1 = 2 and j = 2 then
8:         mat[i − 1][j] ← 1
9:     end if
10:    if i − 1 = 2 and j = 3 then
11:        mat[i − 1][j] ← 1
12:    end if
13: end if
```

B)

```
1: if i − 1 ≠ 0 then
2:     if i = 1 then
3:         if j ≠ 0 and j ≠ 4 then
4:             mat[i][j] ← 1
5:         end if
6:     end if
7:     if i = 2 and j = 2 then
8:         mat[i − 1][j] ← 1
9:     end if
10:    if i = 3 and j = 2 then
11:        mat[i − 1][j] ← 1
12:    end if
13: end if
```

C)

```
1: if i − 1 ≠ 0 then
2:     if i − 1 = 1 then
3:         if j ≠ 1 and j ≠ 5 then
4:             mat[i − 1][j] ← 1
5:         end if
6:     end if
7:     if i − 1 = 2 and j = 2 then
8:         mat[i − 1][j] ← 1
9:     end if
10:    if i − 1 = 3 and j = 2 then
11:        mat[i − 1][j] ← 1
12:    end if
13: end if
```

D)

```
1: if i − 1 ≠ 0 then
2:     if i − 1 = 1 then
3:         if j ≠ 0 and j ≠ 4 then
4:             mat[i − 1][j] ← 1
5:         end if
6:     end if
7:     if i − 1 = 2 and j = 2 then
8:         mat[i − 1][j] ← 1
9:     end if
10:    if i − 1 = 3 and j = 2 then
11:        mat[i − 1][j] ← 1
12:    end if
13: end if
```

Esercizio n°9

Date le seguenti due funzioni:

Pseudocodice esercizio 9

```
1: function MISTERY(a: integer, b: integer) → integer
2:     if 2 × a > b then
3:         return b
4:     else
5:         return a
6:     end if
7: end function
8: function SECRET(a: integer, b: integer) → integer
9:     if a + b > MISTERY(a, b) then
10:        return a
11:    else
12:        return MISTERY(a, b)
13:    end if
14: end function
```

Indicare quale valore RES viene restituito dalla chiamata `secret (24, 3)`.

Esercizio n°10

Dato il seguente pseudocodice:

Pseudocodice esercizio 10

```
1: function FUN(fiore: integer, farfalla: integer) → integer
2:     if fiore = farfalla then
3:         return farfalla
4:     else
5:         if farfalla > fiore then
6:             return FUN(farfalla − fiore, fiore)
7:         else
8:             return FUN(farfalla, farfalla − fiore)
9:         end if
10:    end if
11: end function
```

Cosa si può dire della funzione FUN?

(a) Non termina per alcun valore della coppia (fiore, farfalla)

(b) Non termina soltanto quando fiore = farfalla

(c) Termina sicuramente quando fiore = farfalla

(d) Termina per ogni valore in cui farfalla > fiore

Esercizio n°11

Si considerino le seguenti due funzioni, che prendono in ingresso un numero intero maggiore o uguale a zero:

Pseudocodice esercizio 11

```
1: function EFFE(a: integer) → integer
2:     if a < 2 then
3:         return 2 × a
4:     else
5:         return GI(a − 1) + GI(a − 2)
6:     end if
7: end function
8: function GI(a: integer) → integer
9:     if a < 2 then
10:        return EFFE(a)
11:    else
12:        return EFFE(a − 1) + EFFE(a − 2)
13:    end if
14: end function
```

Indicare quale valore RES viene restituito dalla chiamata `effe(10)`.

Esercizio n°12

Data la seguente funzione:

Pseudocodice esercizio 12

```
1: function CALC(n: integer) → integer
2:     if n = 1 then
3:         return 2
4:     else
5:         return 2 × n − 1 + CALC(n − 1)
6:     end if
7: end function
```

indicare quale tra le seguenti espressioni è il valore che viene restituito se $n \geq 1$.

1. $(n+1)^2$
2. n^2
3. $(n-1)^2$
4. n^{2+1}