Pseudocodice(3.0);

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

Lo Staff delle OII

1 febbraio 2021

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 2020

Quest'anno la selezione scolastica verrà fatta per la prima volta completamente online. Per via di questo cambiamento, ci sono state ovviamente delle modifiche a livello di preparazione dei testi degli esercizi. Una di queste modifiche è stata il ritorno alle regole dell'edizione 2018, ovvero con esercizi di programmazione proposti **esclusivamente in pseudocodice**, contrariamente all'edizione 2019, dove erano proposti sia in pseudocodice che in C/C++.

Reintroduzione del "solo pseudocodice"

Il motivo per cui era stato ripristinato il codice era dovuto alle possibili ambiguità dello pseudocodice, comunque è importante far notare che dalla scorsa edizione sono stati introdotti dei miglioramenti allo pseudocodice utilizzato. È stata eliminata l'ambiguità sulla parola chiave **finché**, ora sostituita con **while**.

Grazie a questi miglioramenti, siamo convinti che quest'anno avere solo lo pseudocodice non creerà problemi per lo svolgimento della selezione scolastica.

Riferimento alla guida delle Olimpiadi del Problem Solving

Come comunicato in precedenza, nei nostri esercizi di programmazione cercheremo di ricalcare per quanto possibile lo pseudocodice già in uso alle OPS. Per riferimento, è possibile consultare la loro guida allo pseudocodice, al seguente indirizzo (comincia da pagina 40):

https://www.olimpiadiproblemsolving.it/site/documenti/20-21/GUIDA_OPS_2020_21.pdf

Lo pseudocodice utilizzato alle selezioni scolastiche delle OII sarà molto simile a quello in uso alle OPS, con piccole differenze puramente estetiche.

Ricordiamo comunque che la sintassi dello pseudocodice non è mai definita in modo esaustivo: è possibile che alcuni esercizi utilizzino dei "nuovi" costrutti. Se questo accadrà, sarà sempre fatto nel modo più facilmente comprensibile. Eventuali "nuovi" costrutti saranno comunque spiegati nel testo dell'esercizio stesso.

Esempi di pseudocodice

Un esempio vale più di mille parole. Nelle pagine seguenti troverete degli esempi di esercizi di precedenti edizioni adattati in pseudocodice.

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, indicizzati (come nel linguaggio C) da w[0] a w[13]
variable mat: char[10][10]	dichiarazione della variabile mat di tipo matrice di caratteri di dimensione 10×10, indicizzati (come nel linguaggio C) da mat[0][0] a mat[9][9]
$a \leftarrow 3$ $a \leftarrow a + 1$	assegnamento del valore 3 ad a, incremento del valore di a
output a	istruzione per scrivere dati sullo schermo
output ''ciao'' **output** ''ciao'', var	scrive la stringa ciao scrive ciao e poi il contenuto della variabile var
array[7] $\leftarrow$ 0 mat[5][28] $\leftarrow$ 1	esempi di inizializzazione di un elemento di un array e di una cella di una matrice
$+ \quad - \quad \times \quad /$ 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** $(a \geq 5)$ $(a \leq 3)$ **and** $(a \leq 5)$ **not** (a)	esempi di operazioni logiche: **not** per il NOT logico, **and** per l'AND, **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

do ... **while** `condizione`	è un ciclo a condizione finale (post-posta), ovvero il corpo viene eseguito fino a che `condizione` diventa FALSA
function CALCOLA(`p1: tipo, ..., pn: tipo`) $\rightarrow$ `tipo` ... **return** `valore_restituito` **end function**	sintassi di una funzione che accetta dei parametri (ciascuno con un tipo) e restituisce un certo valore
procedure ESEGUI(`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

Nota: Mostriamo il codice C/C++ e Pascal in questi esercizi per consentire, a chi ha familiarità con questi due linguaggi, di *confrontarne la sintassi* con quella dello pseudocodice. In gara ci sarà *solo* lo pseudocodice!

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, ' -> ');
  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, " -> "
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 ← RadiceQuadrata(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:
11: function MCM(a: integer, b: integer) → integer
12:     return MYSTER(b, a)
13: 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 di volte per cui viene eseguito il ciclo **while**.

15

Esercizio n°7

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

Pseudocodice esercizio 7

```
1: function F(a: integer[N], 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 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:                    dado ← a[i]
15:                end if
16:            end if
17:        end if
18:        i ← i + 1
19:    end while
20:    return palo + dado
21: end function
```

Indicare quale tra le seguenti affermazioni è FALSA:

(a) La funzione F restituisce 21 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°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:
9: function SECRET(a: integer, b: integer) → integer
10:    if a + b > MISTERY(a, b) then
11:        return a
12:    else
13:        return MISTERY(a, b)
14:    end if
15: end function
```

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

24

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:
9: function GI(a: integer) → integer
10:    if a < 2 then
11:        return EFFE(a)
12:    else
13:        return EFFE(a − 1) + EFFE(a − 2)
14:    end if
15: end function
```

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

110

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 $n \geq 1$.

(a) $(n+1)^2$

(b) n^2

(c) $(n-1)^2$

(d) n^2+1