



Wat is de Informatica Olympiade?

De Nederlandse Informatica Olympiade (NIO) is een programmeerwedstrijd voor de bovenbouw van het Voortgezet onderwijs. Het is een onderdeel van de International Olympiad in Informatics (IOI).

De eerste ronde van de NIO bestaat uit 4 onderdelen A, B, C opgaven en de Codecup (een openbare programmeerwedstrijd):

Opgave	A	B	C	CodeCup
1	5	10	10	
2	10	20	20	
3	15	30	30	
4	25	40	40	
5	25		100	
6	30			
7	40			
8	50			
Totaal	200	100	200	100

Opgaven en links:

- [PDF met alle opgaven van de NIO 2022-2023](#)
- [Digitaal inleverplatform van de NIO](#)
- [Hoofdwebsite van de NIO](#)

De 100 inzendingen met de meeste punten (mits deze minimaal 200 punten hebben gehaald) worden uitgenodigd voor de tweede ronde van de NIO in het voorjaar van 2023 (vorig jaar lag de grens rond de 250 punten, dus vermoedelijk zal het dit jaar vergelijkbaar zijn). Normaliter is de 2^e ronde op de Technische Universiteit Twente (nadere info volgt en is afhankelijk van de corona ontwikkelingen).

Om mee te doen aan de olympiade, moeten je uitwerkingen digitaal zijn ingeleverd bij de NIO voor 20 januari 2023. (Ze worden bij inleveren direct automatisch nagekeken en je kunt tot de deadline zo vaak nieuwe/verbeterde versies insturen als je wilt). De beste deelnemers van de tweede ronde gaan door naar de derde ronde om te bepalen wie er namens Nederland mee mag doen aan de Internationale Informatica Olympiade in de zomer van 2023 in Hongarije.

Inhoud

Wat is de Informatica Olympiade?	1
De PO	3
Puntenverdeling	3
Inleveren en beoordeling	3
Automatisch nakijken	4
Hints en uitleg bij de A opgaven.....	6
Algemene tips.....	6
Opdracht A1: OMDRAAIEN.....	6
Opdracht A2: VAN LEP NAAR LEPEL	7
Opdracht A3: IS HET EEN PALINDROOM	8
Opdracht A4: LETTERS WEGHALEN	8
Opdracht A5: LETTERS TOEVOEGEN.....	8
Opdracht A6: PALINDROMEN ZOEKEN.....	9
Opdracht A7: UNIEKE PALINDROMEN ZOEKEN.....	9
Opdracht A8: ALLE PALINDROMEN ZOEKEN	9
Tips voor de B-opgaven:.....	10
Opmerkingen en tips bij de C-opgaven:	10

De PO

Als handelingsdeel voor Informatica gaan we de een klein deel van de opgaven van de eerste ronde van de NIO maken. Het doel hiervan is om jullie allemaal kennis te laten maken met de olympiade. Je kunt vervolgens zelf kiezen of je een serieuze gooi naar de tweede ronde doet door meer van de opgaven te maken. Dat is voor het handelingsdeel niet verplicht. (Maar je wordt van harte aangemoedigd om dat te doen natuurlijk 🤝). Meer details over hoe je meedoet en welke voorwaarden er zijn, vind je op www.informaticaolympiade.nl. Wt helpt je natuurlijk graag als je officieel mee wilt doen.

Verderop in dit boekje vind je tips en uitleg bij de eerste aantal opgaven, zodat iedereen met deze hulp in staat moet zijn een V(oldoende) te halen voor dit handelingsdeel. In de les is Wt uiteraard ook beschikbaar voor hulp.

De olympiade schrijft niet voor in welke programmeertaal je de oplossingen maakt. De opgaven zijn zo gemaakt dat je in principe in elke taal een oplossing kunt schrijven. In dit document ga ik er van uit dat je het in Python 3 doet. Als je liever een andere taal gebruikt, mag dat natuurlijk ook.

Puntenverdeling

Je doet deze mini-PO alleen of met zijn tweeën. (**Let op:** De deelname aan de NIO zelf is individueel, dus als je graag echt mee wilt doen, doe dan de PO alleen.)

We maken voor de PO alleen opgaven A en eventueel C. Opgaven B worden uniek gegenereerd voor elke deelnemer. Deze opgaven moet je zeker ook maken als je aan de Olympiade mee wilt doen, maar voor de PO slaan we ze over, omdat deze voor Wt niet goed na te kijken zijn.

Deze PO is een handelingsdeel en wordt beoordeeld met een O, V of G. Voor het afronden van het schoolexamen Informatica moet je minimaal een V voor de handelingsdelen halen. De beoordeling van de PO is als volgt:

- Opgaves A1 t/m A5 afgerond en aantoonbaar zelf gemaakt en begrepen: V(oldoende)
- Opgaves A1 t/m A8 en C1 en C2 (of meer) aantoonbaar zelf gemaakt: G(oed)

Inleveren en beoordeling

Je levert bij Wt je gemaakte programma's in. Daarnaast maak je een begeleidend verslagje waarin je de moeilijkste opgave die je gemaakt hebt toelicht (leg uit wat de code die je geschreven hebt doet). Dus als je tot opgave A8 bent gekomen, lever je een documentje in met uitleg van je code van A8.

Let op: deze opdracht is natuurlijk erg plagiaat-gevoelig. Je kunt eenvoudigweg de code van elkaar kopiëren en zo de opgaven maken. Je mag elkaar best helpen, maar zorg dat je begrijpt wat je inlevert. Bij twijfel vraagt Wt een toelichting bij je ingeleverde code om te zien of je begrepen hebt wat je hebt ingeleverd.

Ik heb de drempel van deze PO expres laag gemaakt (t/m A6 is met mijn uitleg uit dit boekje echt prima te doen), dus zorg ook echt dat je het zelf doet/snapt. Elkaar en mij om tips/input vragen mag uiteraard.

Uiterlijke inleverdatum: Vrijdag 14 oktober om 23:59 uur

Automatisch nakijken

De informatica olympiade werkt met een automatisch nakijksysteem. Hierbij kun je een script dat je geschreven hebt als oplossing voor een van de opgaven inleveren. Er worden dan een aantal testcases automatisch op losgelaten en je krijgt punten voor elke testcase waarvoor je script het juiste antwoord geeft. Voor de testcases die misgaan krijg je te zien wat de foutmelding was (bv. Een fout antwoord, een error bij het runnen, maximale runtime overschreden, etc.). Met deze info kun je je script eventueel verbeteren en weer een nieuwe versie indienen. Je kunt elke opgave zo vaak proberen als je wilt, de hoogste score blijft staan.

Zelfs als je niet aan de officiële olympiade wilt meedoen kun je hier dankbaar gebruik van maken om je PO opgaven te testen. Het werkt als volgt:

Ga naar <http://submit.informaticaolympiade.nl/> en maak een account aan (geef Wt even als docent aan).

Als je bent ingelogd zie je links alle A, B en C opgaven staan (en 00 is een testopgave voor als je wilt checken of het inleveren werkt). Klik op de opgave je je wilt inleveren en klik "Voeg inzending toe".

Je kunt nu een script uploaden. Vergeet niet de juiste programmeertaal te selecteren (Python 3):

The screenshot shows the submission interface for the 'Nederlandse Informati' olympiad. The main heading is 'Wedstrijd' (Competition). The participant information is as follows:

Deelnemer	Jochum van Weert
Deelname	Buiten mededinging
Wedstrijd	Eerste ronde NIO 2021
Status	Open voor inzendingen
Jurering	Automatisch
Score (o.v.)	200.00 / 600.00
Tijd tot einde	97 dagen, 14 uren en 29 mi

Below this, there is a 'Nieuws' (News) section with a list of problems: 00 - Som, A1 - Aantal delers, A2 - Iteratie, A3 - De rij voor de bus, A4 - Met het periodiek systeem, and A5 - Totempaal.

On the right side, there is a 'Tijdlimiet (m)' (Time limit) section with 'Jurering' (Judging) set to 'Automatisch', 'Maxscore' (Max score) at 600.00, and 'Score' at 200.00. Below that is a 'Taal' (Language) dropdown menu with 'Python 3' selected. A 'Bestand' (File) field with a 'Choose File' button is also present. At the bottom, there are 'Opslaan' (Save) and 'Annuleren' (Cancel) buttons.

Als je op opslaan klikt wordt je script geüpload en uitgevoerd. Je ziet dan je inzending in de lijst staan met de status “uitvoeren”. Dit betekent dat je script op dit moment uitgevoerd wordt. Als je na een paar seconden de pagina ververs (F5), dan zie je als het goed is de status op “klaar” staan en is er een score gegeven (max 40 voor elke A opgave):

Tijdlimiet (ms)	2000.00
Jurering	Automatisch
Maxscore	40.00
Score	40.00

Tijdstip / Datum	Taal	Status	Score
15:13 /22-09-2020	Python 3	Klaar	40.00

Voeg inzending toe

Je kunt nu op de inzending klikken voor meer info en eventuele foutmeldingen als je niet alle 40 punten hebt gehaald:

Datum / Tijd	2020-09-22 15:13:35
Bestand	Wt_A1.py
Programmeertaal	Python 3
Status	Klaar
Score	40.00 / 40.00

Beschrijving	Feedback	Score	Feedback	Tijd	Score
Voorbeeld	Correct	0.00 / 0.00	Correct	14.0	
Wedstrijd	Correct	40.00 / 40.00	Correct	13.0	
			Correct	10.0	
			Correct	11.0	
			Correct	11.0	
			Correct	15.0	
			Correct	14.0	
			Correct	15.0	
			Correct	14.0	
			Correct	10.0	
			Correct	11.0	

Als er foutmeldingen staan in de rechtertabel, kunnen die je helpen de fouten in je script op te sporen.

De rest van dit document bevat tips, hints en uitleg om je te helpen de eerste 6 A opgaven te tackelen (genoeg voor een V voor de PO)

Hints en uitleg bij de A opgaven

Algemene tips

1. Bij elk van deze opgaven wordt er 1 of meer keer om input gevraagd (bij het automatisch nakijken wordt je programma getest met verschillende inputs). Gebruik hiervoor steeds simpelweg `input()`
De Olympiadesoftware gaat ervan uit dat je precies input leest zoals in de opgave is opgegeven en dat je precies het antwoord geeft zoals is voorgeschreven. Je moet dus geen extra tekst of tussenuitkomsten afdrukken. Doe dus ook niet zoiets als `input("Geef een getal")`, maar simpelweg `input()` (met niets tussen de haakjes), zodat er niets extra's op het scherm wordt geprint. Je opgave wordt anders niet goed gerekend door de nakijksoftware.
2. Om je programma's te testen is het soms vervelend steeds de invoer te moeten intypen. Zet dan tijdens het testen de waardes waar je mee wilt testen rechtstreeks in de code (en commentaar de "input" even weg). Dat scheelt je een hoop typewerk.
Niet vergeten de "input" weer terug te zetten voordat je de opgave instuurt naar de nakijksoftware!
3. De A opdrachten borduren op elkaar voort. Je kunt dus vaak stukken code uit de vorige A opgaven gebruiken om de volgende A opgave op te lossen.

Opdracht A1: OMDRAAIEN

Deze eerste opgave is een echte opwarmer en zou je met slechts enkele regels Python moeten kunnen maken. Je moet het ingevoerde woord achterstevoren zetten en dat weer op het scherm afdrukken. Hiervoor zijn 2 opties (waarvan de eerste optie in Python zeker het makkelijkst is):

1. Gebruik de krachtige slice commando's van python (je weet wel met [en]) om het woord in een keer om te draaien
2. Schrijf een for-loop die langs alle letters van het ingevoerde woord loopt en ze omgedraaid weer aan elkaar zet in een uitvoer-variabele

Tips bij optie 1:

- Bij een string kun je met de slice notatie stukjes van die string aanwijzen en gebruiken.

Bijvoorbeeld:

```
mijn_string = "appeltaart"
print(mijn_string[2:5])
Dit levert de uitvoer "pel" op (letters op positie 2 t/m 4)
```

- Bij het slicen geef je de beginletter en eindletter op en je krijgt dat stukje van de string terug
- Je kunt bij het slicen ook een stapgrootte opgeven. Als je die weglaat is de stapgrootte 1.

Voorbeeld:

```
mijn_string = "appeltaart"
print(mijn_string[0:8:2])
```

Dit levert uitvoer "apla" op, want dat zijn letters 0, 2, 4 en 6 van de string, omdat we nu in stapjes van 2 gaan

- Nu komt de truuk: die stap kan ook negatief zijn en dan loop je als het ware achteruit door de string. Voorbeeld:

```
mijn_string = "appeltaart"  
print(mijn_string[8:1:-1])
```

Dit levert uitvoer "raatlep" op, want dat zijn letters 8 tot en zonder 1 van de string, omdat we nu in stapjes van -1 (dus achteruit) gaan

Je zou nu moeten kunnen bedenken hoe je de hele string omdraait...

Tips bij optie 2:

- Lees de input met `input()` en sla deze op in een variabele
- Maak een variabele met de naam `uitvoer` en maak deze gelijk aan de lege string: `""`
- Maak een for-loop langs alle letters van de invoer (`for letter in invoer:`) en plak de letters een voor een aan de uitvoer string, maar dan aan de voorkant en niet aan de achterkant. Op die manier zet je het woord achterstevoren weer in elkaar
- Print tot slot de opgebouwde uitvoer
-

Voor de lol: superkorte code is nooit een doel op zich, leesbaarheid en logisch nadenken gaat voorop. Toch wel grappig: je kunt deze opdracht oplossen met 1 enkele regel python...

Opdracht A2: VAN LEP NAAR LEPEL

Je kunt hier het beste aan de slag met truuk 1 (het slicen met stapjes van -1) uit opdracht A1. Je draait nu niet het hele woord om, maar alleen alle letters, behalve de laatste. Slim de grenzen van de slice kiezen is daar genoeg voor.

Het resultaat moet je nog even aan het woord van de invoer plakken. In Python kan dat gewoon met + weet je nog?

Voorbeeld:

```
woord1 = "appel"  
woord2 = "taart"  
print(woord1 + woord2)
```

Dit levert een uitvoer van "appeltaart" op

Opdracht A3: IS HET EEN PALINDROOM

Deze opdracht doet hetzelfde als A1, maar je moet even checken met een if of ze aan elkaar gelijk zijn. Eitje toch?

Opdracht A4: LETTERS WEGHALEN

Hier moet je systematisch de mogelijkheden langslopen. Je gaat steeds meer letters weglaten om te kijken of het resulteert in een palindroom. Dat schreeuwt om een loop natuurlijk!

Merk op dat je uiteindelijk altijd op een palindroom uitkomt. Een woord van maar 1 letter is per definitie een palindroom natuurlijk...

Je aanpak zou grofweg als volgt kunnen zijn:

- Lees de input met `input()` en sla deze op
- Maar een teller die bijhoudt hoeveel letters je weg gaat laten. Deze begint op 0 (want je probeert natuurlijk eerst of het meteen een palindroom is, zonder letters weg te laten)
- Maar een oneindige loop met `while True`:
- In de loop check je of het weglaten van "teller" letters aan het einde een palindroom oplevert
- Als dat zo is, onderbreek je de loop met `break` en ben je klaar
- Als dat niet zo is, verhoog `teller`. De loop probeert het nu opnieuw met 1 letter extra weggelaten. En blijft dat dus doen tot hij een palindroom gevonden heeft
- Na de loop print je simpelweg de waarde van `teller`.

Opdracht A5: LETTERS TOEVOEGEN

Deze lijkt op A4, maar dan "andersom".

Merk ook hier op dat als je maar genoeg letters toevoegt, je altijd op een palindroom uitkomt. In het uiterste geval heb je alle letters (behalve de laatste) dubbel en is het altijd een palindroom geworden.

De strategie is vergelijkbaar met A4.

- Lees de input met `input()` en sla deze op
- Maar een teller die bijhoudt hoeveel letters je toe gaat voegen. Deze begint op 0 (want je probeert natuurlijk eerst of het meteen een palindroom is, zonder letters weg te laten)
- Maar een oneindige loop met `while True`:
- In de loop check je of het toevoegen van "teller" letters aan het einde een palindroom oplevert. Denk goed hoe je met slim slicen de juiste letters kiest om toe te voegen.
- Als dat zo is, onderbreek je de loop met `break` en ben je klaar
- Als dat niet zo is, verhoog `teller`. De loop probeert het nu opnieuw met 1 letter extra toegevoegd. En blijft dat dus doen tot hij een palindroom gevonden heeft
- Na de loop print je simpelweg de waarde van `teller`.

Opdracht A6: PALINDROMEN ZOEKEN

Je moet in deze opdracht “door het woord heen wandelen” in steeds groepjes van lengte 5 en checken of dat een palindroom is. Je telt het aantal palindromen dat je zo tegenkomt.

Dit kan het handigst met een for loop. Voorbeeld van een for-loop die alle individuele letters van een woord langsloopt en deze print:

```
for i in range(0, len(woord)):  
    print(woord[i])
```

In deze for loop gebruiken we een variabele *i* die voor ons alle waarden aanneemt tussen 0 en de lengte van het woord. (Let op de bovengrens doet niet mee, dus als het woord 10 letters is, dan neemt *i* alle waarden aan van 0 t/m 9)

Voor elk van die stappen van de loop print hij de letter op positie *i* van het woord. Dit zorgt er dus voor dat hij eerst letter 0 print, dan letter 1, etc.

Pas deze loop aan om deze opgave op te lossen. Nog wat tips:

- Let op dat je niet “te ver” leest. Je gaat slices van lengte 5 pakken. Je kunt niet verder lezen dan het woord lang is, dus je loop moet 5 stappen voor het einde van het woord al stoppen
- Vergelijk in de loop de huidige slice van lengte 5 met zijn eigen omgekeerde versie
- Houd in een teller bij hoe vaak je een palindroom bent tegen gekomen

Opdracht A7: UNIEKE PALINDROMEN ZOEKEN

Deze bouwt voort op A6. Je moet nu alleen ook een lijst bijhouden van alle palindromen die je bent tegengekomen.

- Begin natuurlijk met een lege lijst aan de loop
- Kom je een palindroom tegen, voeg deze aan de lijst toe
- Voordat je een palindroom toevoegt, check je of hij al in de lijst zit. Dan tel je hem niet mee

Relevante commando's

```
#voegt een nieuwe string toe aan een bestaande lijst:  
mijn_lijst.append("nieuw item voor in de lijst")
```

```
#checkt of een waarde in een lijst voorkomt:  
if ("appeltaart" in mijn_lijst):
```

Opdracht A8: ALLE PALINDROMEN ZOEKEN

Deze bouwt weer verder op A7. Je zult nu in je for-loop die langs alle posities van de input loopt en checkt of hij palindromen ziet een for-loop moeten toevoegen (dus een loop in een loop) die voor elke positie (buitenste loop) elke mogelijke lengte palindroom (binnenste loop) checkt.

Ook hier houd je al gevonden palindromen in een lijst bij en telt dubbele zo niet mee.

Tips voor de B-opgaven:

Voor deze PO hoeft het niet, maar voor een serieuze deelname aan de Olympiade moet je natuurlijk ook de B-opgaven maken. Deze werken wat anders dan de A- en C-Opgaven. Je krijgt namelijk een speciaal voor jou gegenereerde “puzzel” waarvoor je alleen maar het juiste antwoord hoeft in te typen. Hoe je aan dat antwoord komt, maakt voor de olympiade niet uit. In de praktijk zul je een (python) script schrijven om tot het antwoord te komen, maar als je bijvoorbeeld handig bent met Excel kun je het soms ook daar mee oplossen.

Een paar dingen om op te letten:

- Je kiest zelf wanneer de opgave voor jou gegenereerd wordt (je krijgt echt een unieke versie van de “puzzel”)
- Zodra jouw opgave gegenereerd is, heb je een week om het antwoord te geven. Daarna krijg je strafpunten voor elke dag te laat
- Als je een fout antwoord geeft krijg je ook strafpunten. Je kunt het uiteraard nog eens proberen, maar je kunt niet gokken, want dan ben je zo al je punten kwijt

Vanwege bovenstaande dingen moet je dus op een geschikt moment de opgave genereren, omdat de timer dan gaat lopen. Ook is het verstandig je antwoord goed te dubbelchecken om strafpunten te voorkomen.

Bij de meeste B-opgaven is het handig om een python script te schrijven om tot de oplossing te komen. Hiervoor is het handig als je script de puzzel kan inladen uit een textfile (die krijg je van de olympiade bij je opgave). Hiervoor kun je het volgende voorbeeldscript gebruiken (het commentaar in de code legt uit wat er gebeurt:

```
#open de tekstfile om hem te lezen (vul zelf de goede bestandsnaam in):
with open("mijntextfile.txt") as f:
    #lees de inhoud van de file in een variabele
    #en haal eventuele spaties op het einde weg met strip():
    inhoud = f.read().strip()
    #splits de invoer per regel in een lijst met elementen
    regels = inhoud.split("\n")

#print de lijst om te checken of het gelukt is:
print(regels)
```

Opmerkingen en tips bij de C-opgaven:

De C opgaven zijn vergelijkbaar met de A-Opgaven, maar ze zijn gewoon een tandje moeilijker. Wt heeft hier geen handleiding voor je. Ik denk natuurlijk graag met je mee en geef tips als je daarom vraagt.

Succes!