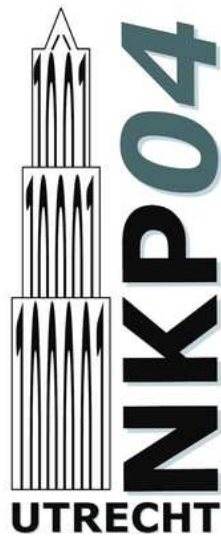


Teamhandleiding

Nederlands Kampioenschap Programmeren 2004

28 november 2004



Inhoudsopgave

1	Samenvatting	3
1.1	Inlezen en wegschrijven	3
1.2	Programmeertalen	3
1.3	Insturen van oplossingen	3
1.4	Bekijken van scores, inzendingen, e.d.	3
2	De computers	4
3	Opgaven	4
4	Programmeertalen	5
4.1	C & C++	5
4.2	Java	5
4.3	Pascal	6
4.4	Haskell	6
5	Het jurysysteem: DOMjudge	7
5.1	Inleiding	7
5.2	Oplossingen insturen: <code>submit</code>	7
5.3	De uitslag bekijken van inzendingen	7
5.3.1	Mogelijke uitslagen	8
5.4	Clarification requests insturen	8
5.5	Hoe worden opgaven beoordeeld?	8
5.5.1	Insturen	9
5.5.2	Compileren	9
5.5.3	Testen	9
5.5.4	Beperkingen	9
A	Code voorbeelden	11

1 Samenvatting

Hieronder staat kort samengevat de belangrijkste informatie. Dit is bedoeld om snel aan de slag te kunnen. We raden echter ten zeerste aan, dat minstens iemand binnen je team de complete handleiding doorneemt, omdat daarin specifieke details van de verschillende programmeertalen en het jurystelsel staan, die ook van belang kunnen zijn.

1.1 Inlezen en wegschrijven

Tijdens de wedstrijd is het de bedoeling dat alles wat je inleest, van de ‘standard in’ (toetsenbord) gelezen wordt, en weggeschreven wordt naar de ‘standard out’ (beeldscherm). Je hoeft dus nooit een bestand te openen. Zie voor voorbeeldcode de bijlage van de systeemhandleiding.

Let op dat bij wegschrijven het gebruik van hoofd- en kleine letters verschil maakt. Dit betekent dat als je iets wegschrijft, het er toe doet of dat met hoofdletters is, of met kleine letters. Ook extra spaties zorgen ervoor dat de jurycomputer je oplossing afkeurt.

1.2 Programmeertalen

De volgende programmeertalen kunnen gebruikt worden: C, C++, Java, Pascal en Haskell. Voor de exacte compiler-opties die we gebruiken en taal-specifieke details wordt verwezen naar de complete team-handleiding.

1.3 Insturen van oplossingen

Insturen van oplossingen gaat via het command-line programma `submit`. Gebruik `submit <probleem>.<extensie>`, met de letter van het probleem en een standaard extensie van je taal. Voor de complete documentatie en alle opties, zie `submit --help`.

1.4 Bekijken van scores, inzendingen, e.d.

Het bekijken van inzendingen, scores en sturen en lezen van “clarification requests” gaat via een webinterface. Het adres is <http://judge.nkp.nl> en de pagina wijst zich verder vanzelf.

2 De computers

Alle teams krijgen één computer aangewezen, waarop zij kunnen werken om de opgaven op te lossen. Dit zijn computers van het Informatica Instituut en zijn allemaal Dell computers met 15 inch TFT flatscreens met een Intel Pentium 3 of 4 processor van minimaal 1 GHz en 256 MB intern geheugen.

Tijdens het NKP kan gebruik gemaakt worden van twee verschillende besturingssystemen: Microsoft Windows 2000 Professional of CentOS GNU/Linux (RedHat Enterprise). Beide draaien op dezelfde hardware. Je keuze voor Windows of Linux moet je van tevoren aangeven.

3 Opgaven

Voor iedere opgave is het noodzakelijk om de invoer van het probleem in te lezen en het antwoord weer uit te voeren. Dit moet gedaan worden via “standaard invoer en uitvoer”. Je programma hoeft niet (en mag ook niet) zelf bestanden openen, maar kan gewoon van de “console” lezen en schrijven. Je programma mag wel naar “standard error” schrijven, maar dit wordt genegeerd tijdens het jureren.

Verder bestaat de invoer van opgaven standaard uit meerdere testgevallen. Als eerste is dan op één regel het aantal testgevallen gegeven en daarna volgen de testgevallen.

Zie verder bijlage A voor voorbeelden van code om dit af te handelen.

4 Programmeertalen

Tijdens het NKP kan gebruikt gemaakt worden van de programmeertalen C, C++, Java, Pascal en Haskell. Je mag zelf weten in welke taal je een oplossing instuurt: je mag verschillende opgaven in verschillende talen proberen op te lossen en zelfs voor één opgave oplossingen in verschillende talen insturen.

Voor iedere taal zijn er afhankelijk van het platform waaronder je werkt, verschillende compilers en IDE's aanwezig. De compiler die wij gebruiken om je programma te compileren, staat aangegeven in tabel 2.

Hieronder staat per taal beschreven, welke compilers er beschikbaar zijn en andere taalspecifieke details. Verder een beknopte tabel (1) met alle beschikbare compilers en welke versies.

Tabel 1: Beschikbare compilers

Compiler	Linux versie	Windows versie
GCC	3.2.3	3.3.3
MS Visual C++		7.1
GCJ	3.2.3	3.3.3
Sun Java	1.4.2	1.4.2
FreePascal	1.0.10	1.0.10
GPC	NOTINSTALLED	3.3.3
GHC	6.2.1	6.2.1
HUGS	November 2003	December 2001

4.1 C & C++

Onder Windows is GCC (de GNU C/C++ compiler collectie) beschikbaar via Cygwin.

In C++ kan gebruik gemaakt worden van de Standard Template Library (STL).

GCC ondersteunt de volledige ANSI C89 en C++ standaarden en GNU extensies. Windows gebruikers moeten opletten, dat het type van een 64-bits integer `long long` is en dat alle STL datatypes binnen de namespace `std` gedeclareerd zijn.

4.2 Java

Voor Java gebruiken wij niet de Java compiler van Sun zelf, maar de compiler van GNU: GCJ versie 3.2.3 (<http://gcc.gnu.org/java/>). GCJ ondersteunt alle delen van Java, die nodig zijn tijdens het NKP (niet relevante delen, zoals `java.swing` en `java.awt` worden niet (volledig) ondersteund).

De Java compiler van Sun zelf is ook beschikbaar en verder zullen de JavaDocs beschikbaar zijn. Onder Windows staat er een link in het startmenu en onder Linux is de link `file:///usr/local/javadocs/index.html`.

Let op: de hoofdklasse van je programma moet `Main` (met hoofdletter) heten en een public function `main` bevatten! De klasse zelf moet juist *niet* public of private gedeclareerd zijn. Zie de voorbeeldcode in de appendix.

4.3 Pascal

Voor Pascal gebruiken wij de FreePascal compiler, versie 1.0.10 (<http://www.freepascal.org/>). Deze is onder Linux en Windows beschikbaar. De bijbehorende IDE is niet beschikbaar.

Verder is GNU GPC beschikbaar onder Windows via cygwin.

Let op dat het datatype `integer` de 16 bits `smallint` is. Verder ondersteunt de compiler de ANSI Pascal standaard plus nagenoeg alle Turbo Pascal uitbreidingen. Labels en `goto` worden ook ondersteund.

4.4 Haskell

Waarschuwing vooraf: Haskell is een functionele programmeertaal en verschilt daarmee erg van de andere (imperatieve) talen. We geven geen garanties dat voor alle opgaven in Haskell een oplossing (binnen de tijdslimiet) bestaat! Verder kan Haskell gewoon gebruikt worden.

Als compiler gebruiken we de Glasgow Haskell Compiler, GHC (<http://www.haskell.org/ghc/>) versie 6.2.1. Deze compiler ondersteunt de volledige Haskell 98 standaard.

GHC en ook HUGS zijn beide beschikbaar onder Linux en Windows.

5 Het jurysysteem: DOMjudge

5.1 Inleiding

Dit hoofdstuk beschrijft de onderdelen van het DOMjudge jurysysteem, die aan de teams ter beschikking staan om oplossingen in te sturen, uitslagen te bekijken en “clarification requests” te sturen.

Het systeem is voor een groot deel gebaseerd op een webinterface. Het insturen van oplossingen gaat echter via een command-line programma.

5.2 Oplossingen insturen: submit

Het insturen van oplossingen voor problemen gebeurt via een command-line interface: het programma `submit` (onder Windows `submit.exe`).

Syntax: `submit [opties] bestandsnaam.ext`

Het `submit` programma haalt de naam van het probleem uit `bestandsnaam` en de de programmeertaal uit de extensie `ext`. Dit kan handmatig aangepast worden met de opties `-p probleemnaam` en `-l taalextensie`. Zie `submit --help` voor een compleet overzicht van mogelijke opties en extensies en een aantal voorbeelden. Let op, dat deze help-tekst redelijk lang is en misschien niet op één scherm past. Gebruik dan bijvoorbeeld `submit --help | more` om alles te lezen.

`submit` zal het bestand controleren op een aantal eigenschappen en eventueel waarschuwingen geven, zoals wanneer het bestand al lange tijd niet veranderd is of groter is dan de maximale source-code grootte.

Hierna geeft `submit` een kort overzicht met de details van de inzending en vraagt om bevestiging. Controleer vooral of je het goede bestand, probleem en taal hebt en druk dan op ‘y’ om de oplossing in te sturen. Als alles goed gaat, zal `submit` een melding geven dat de inzending succesvol is. Indien niet, zal er een foutmelding verschijnen.

Het `submit` programma maakt gebruik van een directory `.submit` in de homedirectory van het account (`cygwin\.submit` onder Windows). Hier slaat het tijdelijk bestanden op voor inzending en staat ook een logfile `submit.log`. Verwijder deze directory niet of pas hem niet aan, omdat anders het `submit` programma eventueel niet meer correct functioneert. Verder is een “public ssh-key” van de jury in de ssh configuratie toegevoegd. Deze is ook nodig voor het functioneren van `submit`.

5.3 De uitslag bekijken van inzendingen

Op de team webpagina staat een overzicht van je inzendingen. Dit overzicht bevat alle relevante gegevens: de tijd van inzending, de programmeertaal, het probleem en de status. Het adres van je teampagina is <http://judge.nkp.nl>. Hiervandaan kun je ook naar de publieke pagina, waarop de scores van alle teams vermeld staan.

5.3.1 Mogelijke uitslagen

Op een ingestuurde oplossing zijn een aantal verschillende uitslagen mogelijk. Hier volgen ze, met een korte beschrijving:

CORRECT	Je oplossing was goed en je hebt dit probleem opgelost!
COMPILER-ERROR	Het compileren van je programma gaf een fout. Zie de inzendingsdetails voor de precieze foutmelding. <i>Compilerwaarschuwingen</i> worden niet als fouten gezien, maar kun je eventueel wel bekijken bij de details van je inzending.
TIMELIMIT	Je programma draaide langer dan de maximaal toegestane tijd en is afgebroken. Dit kan betekenen dat je programma ergens in een loop blijft hangen, of dat je oplossing niet efficiënt genoeg is.
RUN-ERROR	Je programma gaf een fout tijdens het uitvoeren. Dit kan verschillende oorzaken hebben, zoals deling door nul, incorrecte geheugen adressering (segfault, bijvoorbeeld door arrays buiten bereik te indiceren), te veel geheugengebruik, enzovoort... Let ook op, dat je programma met een exitcode 0 eindigt!
NO-OUTPUT	Je programma gaf geen uitvoer. Let op dat je uitvoer naar standard output schrijft!
WRONG-ANSWER	De uitvoer van je programma was niet correct. Het kan zijn dat je oplossing niet correct is, maar let ook goed op dat je de antwoorden precies zoals beschreven uitvoert: de uitvoer moet exact kloppen met de antwoorden van de jury!

5.4 Clarification requests insturen

Communicatie met de jury loopt door middel van clarifications, deze komen op je teampagina te staan. Boven aan de pagina de gedane clarifications, daar onder je requests. Je kan vragen aan de jury stellen door middel van het doen van een “Clarification Request”, de link hiervoor bevindt zich onderaan de clarifications-pagina. Je vraag zal alleen bij de jury aan komen, zij zullen deze zo snel mogelijk en adequaat beantwoorden. Antwoorden die voor iedereen relevant zijn zullen naar iedereen gestuurd worden.

5.5 Hoe worden opgaven beoordeeld?

Het DOMjudge jurysysteem is volledig geautomatiseerd. Dit betekent dat er (in principe) geen menselijke interactie is tijdens de beoordeling. Het beoordelen gebeurt via de volgende stappen:

5.5.1 Insturen

Via het `submit` (zie 5.2) programma kun je een oplossing voor een opgave insturen. Let op dat je de source-code van je programma moet insturen (en dus niet een gecompileerd programma of de uitvoer van je programma). `submit` maakt dan een verbinding met een server en die kopieert je programma dan naar de jury-computers.

Dan komt je programma in de wachtrij te staan, om gecompileerd, uitgevoerd en de uitvoer getest te worden op één van de jury-computers.

5.5.2 Compileren

Je programma wordt op een jury-computer onder Linux gecompileerd. De exacte compilers en opties die we gebruiken, staan in tabel 2. Als je een andere compiler of besturingssysteem gebruikt, moet dat in principe geen probleem zijn, maar let wel op dat je geen compiler/systeem-specifieke dingen gebruikt (bij een compileerfout kun je altijd de foutmelding bekijken).

Tabel 2: Compiler commando's

programmeertaal	jury compiler-commando	niet-statisch gelinkt
C	<code>gcc -Wall -O2 -static -lm</code>	<code>gcc -Wall -O2 -lm</code>
C++	<code>g++ -Wall -O2 -static</code>	<code>g++ -Wall -O2</code>
Java	<code>gcj -Wall -O2 -static --main=Main</code>	<code>gcj -Wall -O2 --main=Main</code>
Pascal	<code>fpc -viwn -O2 -Sg -XS</code>	<code>fpc -viwn -O2 -Sg</code>
Haskell	<code>ghc -Wall -O -static -optl-static</code>	<code>ghc -Wall -O</code>

Merk op dat wij statisch compileren i.v.m. het jurysysteem, maar wat betreft foutmeldingen e.d. zijn de compiler-commando's in de kolom "niet-statisch gelinkt" equivalent.

5.5.3 Testen

Als je programma succesvol gecompileerd is, wordt het gedraaid en de uitvoer vergeleken met de correcte uitvoer van de jury. Er wordt eerst gecontroleerd of je programma correct geëindigd is: als je programma met een fout eindigt en het goede antwoord geeft, krijg je toch een `run-error!` Er zijn een aantal beperkingen die aan je programma opgelegd worden. Als je programma die overschrijdt, wordt het ook afgebroken met een fout, zie (5.5.4).

Verder moet de uitvoer van jouw programma exact overeenkomen met de uitvoer van de jury. Let dus goed op, dat je de uitvoerspecificatie volgt.

5.5.4 Beperkingen

Om hacken tegen te gaan, het jurysysteem stabiel te houden en iedereen duidelijke, gelijke omstandigheden te geven, zijn er een aantal beperkingen die aan iedere ingestuurde oplossing opgelegd worden:

compile-tijd Je programma mag er maximaal 30 seconden over doen om te compileren. Daarna wordt het compileren afgebroken en levert dit een compileerfout

op. Dit zou in de praktijk nooit een probleem mogen opleveren. Mocht dit toch gebeuren bij een normaal programma, laat het dan de organisatie weten.

- source grootte** De sourcecode van je programma mag maximaal 256 kilobytes groot zijn. Bij het inzenden zul je hier al voor gewaarschuwd worden, bij compileren levert dit ook een compileerfout op.
- geheugen** Je programma heeft tijdens het draaien maximaal 65536 kilobytes geheugen ter beschikking. Let op dat dit totaal geheugen is (inclusief programma-code, statisch en dynamisch gedefiniëerde variabelen, stack, ...)! Als je programma meer probeert te gebruiken, zal het waarschijnlijk afbreken, zodat dit een “RUN-ERROR” geeft.
- bestands grootte** Hoewel je geen bestanden mag schrijven, wordt de standaard (fout) uitvoer (stdout/stderr) wel naar een bestand geschreven. De maximale grootte van deze bestanden is 4096 kilobytes (per bestand).
- aantal processen** Het is niet de bedoeling dat je programma meerdere processen (threads) start. Dit heeft ook geen zin, want je programma heeft precies 1 processor volledig tot zijn beschikking. Om de stabiliteit van het jurysysteem te bevorderen, kun je maximaal 8 processen tegelijk draaien (inclusief de processen waardoor je programma gestart is).
- Mensen die nooit met meerdere processen geprogrammeerd hebben (of niet weten wat dat is), hoeven zich geen zorgen te maken: standaard draait een gecompileerd programma in één proces.

A Code voorbeelden

Hieronder staan een aantal voorbeelden van code om de invoer van een probleem in te lezen en de uitvoer weg te schrijven.

De code hoort bij de volgende probleembeschrijving: gegeven een string met een naam (één woord), voer de string “Hallo <naam>!” uit.

Dit probleem zou de volgende in- en uitvoer kunnen hebben:

Invoer	Uitvoer
3 wereld Jan Sinterklaas	Hallo wereld! Hallo Jan! Hallo Sinterklaas!

Let op dat het op de eerste regel het getal 3 aangeeft, dat er 3 testgevallen volgen.

Een oplossing voor dit probleem in C:

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int aantaltests, test;
6      char naam[100];
7
8      scanf("%d\n",&aantaltests);
9
10     for(test=1; test<=aantaltests; test++) {
11         scanf("%s\n",naam);
12         printf("Hallo %s!\n",naam);
13     }
14
15     return 0;
16 }
```

Let op de `return 0;` aan het einde, zodat we geen run-error krijgen!

Een oplossing in C++ kan als volgt:

```
1  using namespace std;
2
3  #include <iostream>
4  #include <string>
5
6  int main()
7  {
8      int aantaltests, test;
9      string naam;
10
11     cin >> aantaltests;
12
13     for(test=1; test<=aantaltests; test++) {
14         cin >> naam;
15         cout << "Hallo " << naam << "!\n";
16     }
17
18     return 0;
19 }
```

Een oplossing in Pascal:

```
1  program voorbeeld(input, output);
2
3  var
4      aantaltests, test : integer;
5      naam                : string[100];
6
7  begin
8      readln(aantaltests);
9
10     for test := 1 to aantaltests do
11         begin
12             readln(naam);
13             writeln('Hallo ',naam,'!');
14         end;
15     end.
```

En tenslotte een oplossing in Java:

```
1  import java.io.*;
2
3  class Main {
4      public static BufferedReader in;
5
6      public static void main(String[] args) throws Exception {
7          int aantaltests, test;
8          String naam;
9
10         in = new BufferedReader( new InputStreamReader(System.in) );
11
12         aantaltests = Integer.parseInt(in.readLine());
13
14         for(test=1; test<=aantaltests; test++) {
15             naam = in.readLine();
16             System.out.print("Hallo "+naam+"\n");
17         }
18     }
19 }
```