jBACI – Käyttöohje

(Tätä ohjetta saa käyttää yliopiston opetustarkoituksiin ja jatkokehittelyyn.)

1. Yleistä

jBACI on työkalu, jota käytetään rinnakkaisuuden hallinnan havainnollistamiseen ja harjoitteluun. Se on rinnakkaisuuden simulaattori, jolla voidaan ohjelmaa suorittaessa tutkia sen eri prosesseja ja niiden suoritusjärjestystä askel askeleelta.

2. Asennusohjeet

jBACI tarvitsee toimiakseen Java (SDK tai JRE) version 1.4. Asenna tämä koneellesi, jos sinulla ei sitä vielä ole.

2.1. Asennus Windowsiin

- Lataa jBACIn versio 1.4.5 koneellesi. Tiedosto löytyy esimerkiksi osoitteesta: http://stwww.weizmann.ac.il/g-cs/benari/jbaci/jbaci1-4-5.zip.
- Pura zip -tiedosto haluamaasi kansioon.
- Avaa config.cfg tiedosto esimerkiksi notepadiin ja muokkaa oikeat polut seuraaville riveille:
    - SOURCE_DIRECTORY (examples-kansio)
    - PASCAL_COMPILER (bapas.exe)
    - C_COMPILER (bacc.exe)
- Bapas.exe ja bacc.exe löytyvät kansiosta bin. Ole erityisen huolellinen sen suhteen, että polkuihin tulee kaksoiskenoviivat tavallisen yhden viivan sijaan.
- Avaa run.bat käynnistääksesi ohjelman tai Windowsissa voit myös avata suoraan jbaci.jar –tiedoston.

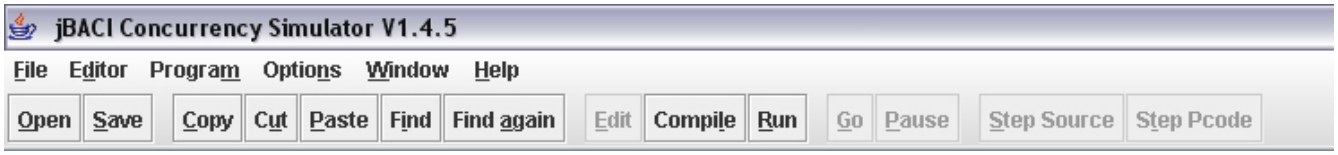
2.2. Asennus Linuxiin

- Lataa ja pura jBACI aluksi Windows-ohjeiden mukaisesti
- Lataa seuraava tiedosto http://www.mines.edu/fs_home/tcamp/baci/balnxxe-2007Jun02.tar.gz

- Pura se jBACIn bin-kansioon. Voit halutessasi poistaa tästä kansiosta entuudestaan löytyvät bacc.exe ja bapas.exe. Ne korvataan balnxxe-kansiosta löytyvillä bacc- ja bapas-tiedostoilla.
- Avaa config.cfg tiedosto esimerkiksi notepadiin ja muokkaa oikeat polut seuraaville riveille:
    - SOURCE_DIRECTORY (examples-kansio)
    - PASCAL_COMPILER (bapas balnxxe-kansiossa)
    - C_COMPILER (bacc balnxxe-kansiossa)
- Bapas.exe ja bacc.exe löytyvät kansiosta bin.
- Avaa komentotulkki, mene kansioon, johon asensit jBACIn, ja kirjoita komennoksi *java –jar jbaci.jar*

3. jBACIn käyttö

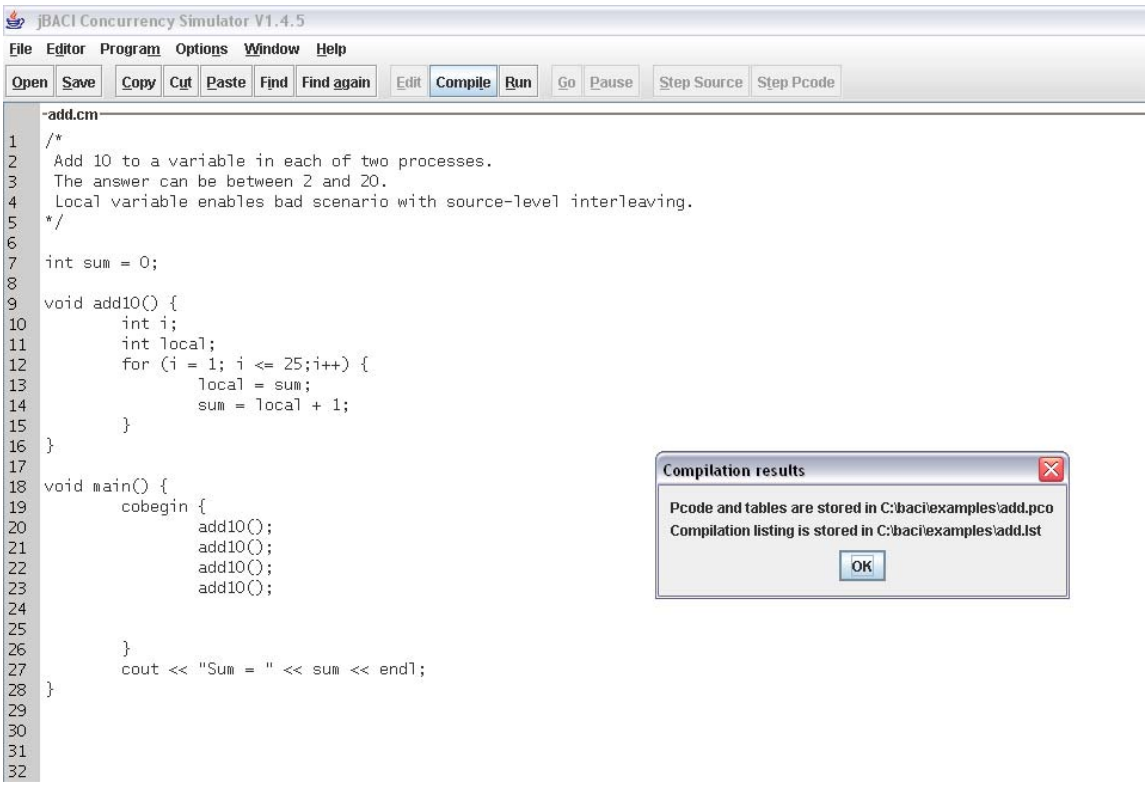
Käynnistettyäsi jBACIn, näet ohjelman valikot ja myös tärkeimmät toiminnot sisältävät napit valikoiden alla. Nappien toiminnot löytyvät myös valikoista.



Kuva 1: jBACIn valikot perustilassa

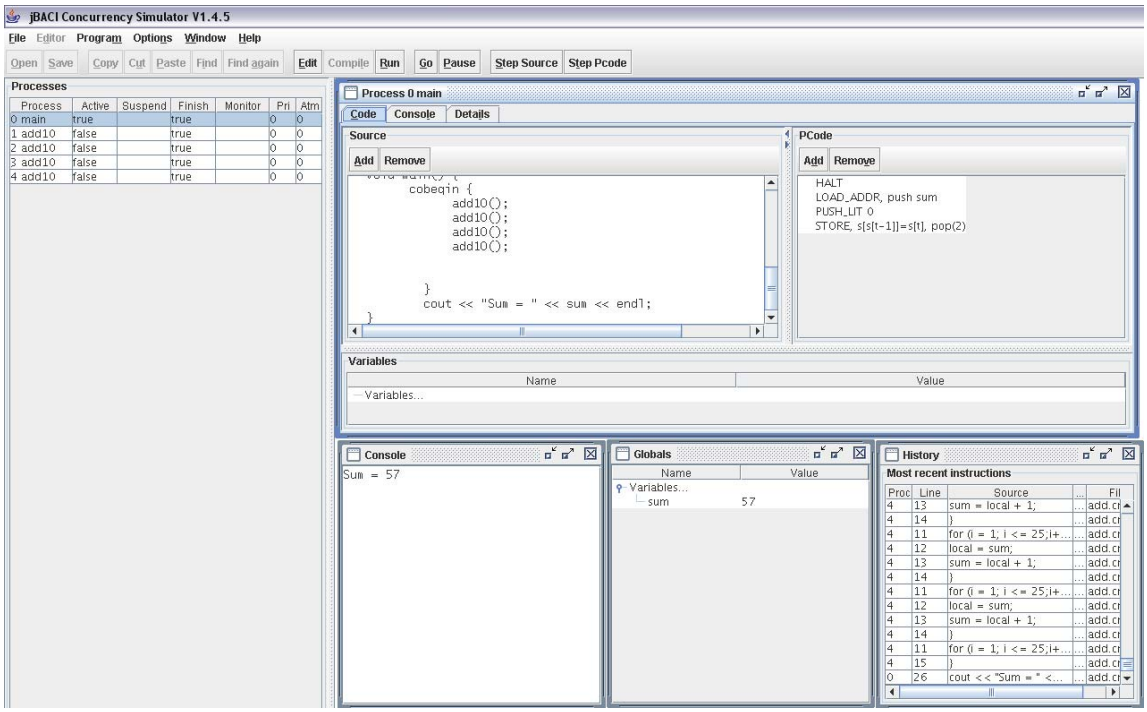
jBACIa voi käyttää joko editointi- tai suoritustilassa. Riippuen tilasta, valikon toiminnot ovat joko käytettävissä tai poissa käytöstä. Esimerkiksi editointitilassa toimivat vain ne toiminnot, joita voi käyttää editointiin ja suoritustilassa suoritukseen. Ikkunat näkyvät vain suoritustilassa.

Editointitilassa näytetään lähdekoodi, jota voi muokata. Lähdetiedostojen on oltava .cm tai .pm -päätteisiä (eli koodi on oltava kirjoitettu C-- tai Pascal-kielellä), jolloin kääntäjä osaa valita automaattisesti oikean kääntäjän. jBACI näyttää käännöksen tuloksen pop up -ikkunassa. Jos koodissa on virheitä, kursori siirtyy ensimmäisen virherivin alkuun.



Kuva 2: jBACI editointitilassa, käännös suoritettuna

Kun ohjelma on käännetty oikein, pääsee sitä tarkastelemaan suoritustilaan. Suoritustila on jaettu kahteen osaan: vasemmalla on prosessitaulu ja oikealla ikkunat. Prosessitaulu näyttää jokaisen prosessin omalla rivillään ja tiedot kyseisestä prosessista.



Kuva 3: jBACI suoritustilassa

Tutustumme aluksi toimintoihin, joita valikot ja ikkunat sisältävät, jonka jälkeen käymme läpi jBACIn toimintaa tarkemmin esimerkin avulla.

3.1. Valikot

3.1.1. File-valikko

File-valikosta löytyvät perustoiminnot, joista suurin osa toimii ainoastaan editointitilassa:

New	Aukaisee uuden tyhjän tiedoston
Open	Aukaisee jo olemassa olevan tiedoston
Save	Tallettaa tiedoston
Save as	Tallettaa tiedoston uudella nimellä
Exit	Sulkee ohjelman

3.1.2. Editor-valikko

Editor-valikosta löytyvät toiminnot lähdekoodin muokkaamiseen:

Copy	Kopioi
Cut	Leikkaa
Paste	Liitä
Find	Etsii sanan tai lauseen koodista
Find again	Etsii uudelleen saman sanan tai lauseen

3.1.3. Program-valikko

Program-valikosta löytyvät toiminnot editointi- ja suoritustilojen käynnistämiseen:

Edit	Vaihtaa suoritustilasta editointitilaan
Compile	Kääntää lähdekoodin
Run	Vaihtaa editointitilasta suoritustilaan

Lisäksi suoritustilassa tästä valikosta löytyvät myös seuraavat toiminnot:

Go	Aloittaa ohjelman suorituksen
Pause	Pysäyttää ohjelman suorituksen
Step source	Suorittaa lähdekoodia askel kerrallaan
Step pcode	Suorittaa konekäskyjä askel kerrallaan

3.1.4. Options-valikko

Options-valikossa voi seuraavia asetuksia ottaa käyttöön tai poistaa käytöstä:

Pause on Process Swap	Tämän ollessa valittuna, ohjelman suoritus pysähtyy aina automaattisesti, kun prosessia vaihdetaan.
Show Active Window	Tämän ollessa valittuna, näkyy aktiivisen prosessin ikkuna etummaisena koodia suorittaessa askel kerrallaan. Jos asetus ei ole valittuna, ei eri prosesseille avaudu omaa ikkunaa suorituksen aikana.

History of Source Steps	Tämän ollessa valittuna, näytetään *historia*-ikkunassa vain lähdekoodin historia. Muulloin näytetään sekä lähdekoodin että konekäskyjen historia.
Write History File	Tämän ollessa valittuna, talletetaan koko suoritetun lähdekoodin tai kaikkien suoritettujen konekäskyjen historia tiedostoksi. his -päätteinen tiedosto talletetaan samaan kansioon kuin alkuperäinen lähdekoodi.

3.1.5. Windows-valikko

Window-valikosta voi avata eri ikkunoita, kun ohjelma on suoritustilassa. Lisätietoa eri ikkunoista löytyy kohdasta: 3.2. Ikkunat.

3.1.6. Help-valikko

Help-valikosta löytyy tietoja ohjelman versiosta, sen tekijöistä yms.

3.2. Ikkunat

Kun ohjelma on suoritustilassa, avautuu työpöydälle uusia ikkunoita. Ikkunoita pystyy siirtämään hiirellä ja muokkaamaan niiden kokoa haluamakseen.

3.2.1. Process

Prosess-ikkuna on jBACIn ydin. Se sisältää kolme alavalikkoa: C*ode*, *Console* ja D*etails*.

- *Code*-valikko näyttää ohjelman koodin.
  - *Source* -osassa näytetään ohjelman koko lähdekoodi, ei vain yhden prosessin koodia.
  - *Pcode* -osassa näytetään ohjelman konekäskyt, mutta vain valitulta lähdekoodin riviltä, eikä siis koko lähdekoodista. Pcode-osa näkyy ainoastaan, jos Prosess-ikkunan kokoa kasvattaa manuaalisesti.
  - *Variables* -osassa näytetään prosessin paikallisten muuttujien arvot.

- *Console*-valikko näyttää, mitä juuri kyseinen prosessi tulostaa, toisin kuin erillinen *Console*-ikkuna (ks. kohta 3.2.2), jossa näkyy koko ohjelman tulostus.
- *Details-ikkuna* näyttää prosessipinojen sisällön sekä prosessien tilan.

3.2.2. Console

Console-ikkuna näyttää, mitä ohjelma tulostaa.

3.2.3. Globals

Globals-ikkuna näyttää globaalien muuttujien arvot.

3.2.4. History

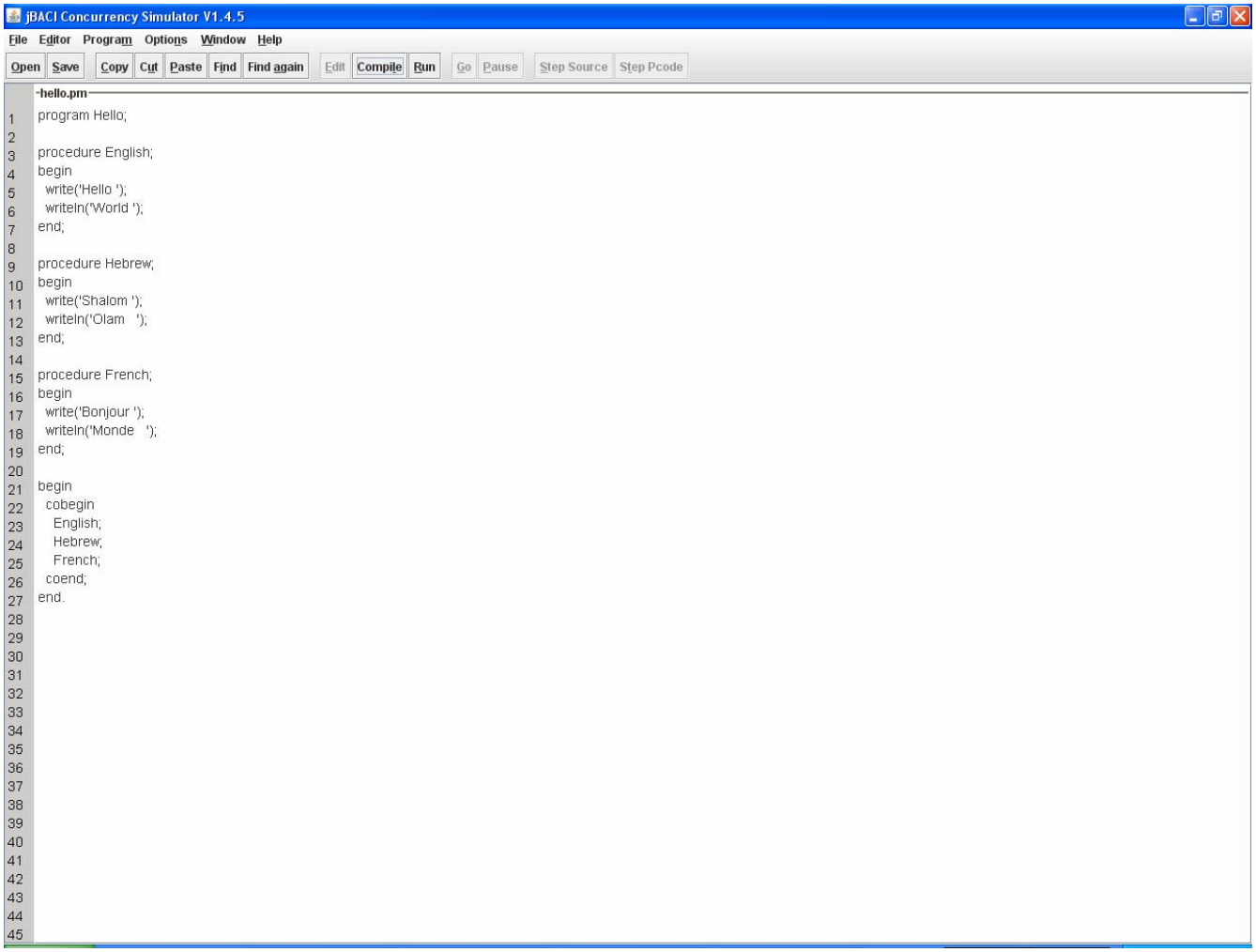
Historia-ikkuna näyttää viimeiset 150 lähdekoodin suoritusta.

3.2.5. Linda Board

Linda on jBACIn laajennus, jonka on kehittänyt David Gelernter. Tätä ikkunaan ei tarvita Rinnakkaisohjelmointi –kurssilla, jota varten tämä ohje on tehty. Lisätietoja *Lindasta* löytyy seuraavasta linkistä, josta voit ladata jBACI viralliset ohjeet:
http://stwww.weizmann.ac.il/g-cs/benari/jbaci/jbaci1-4-5docs.zip

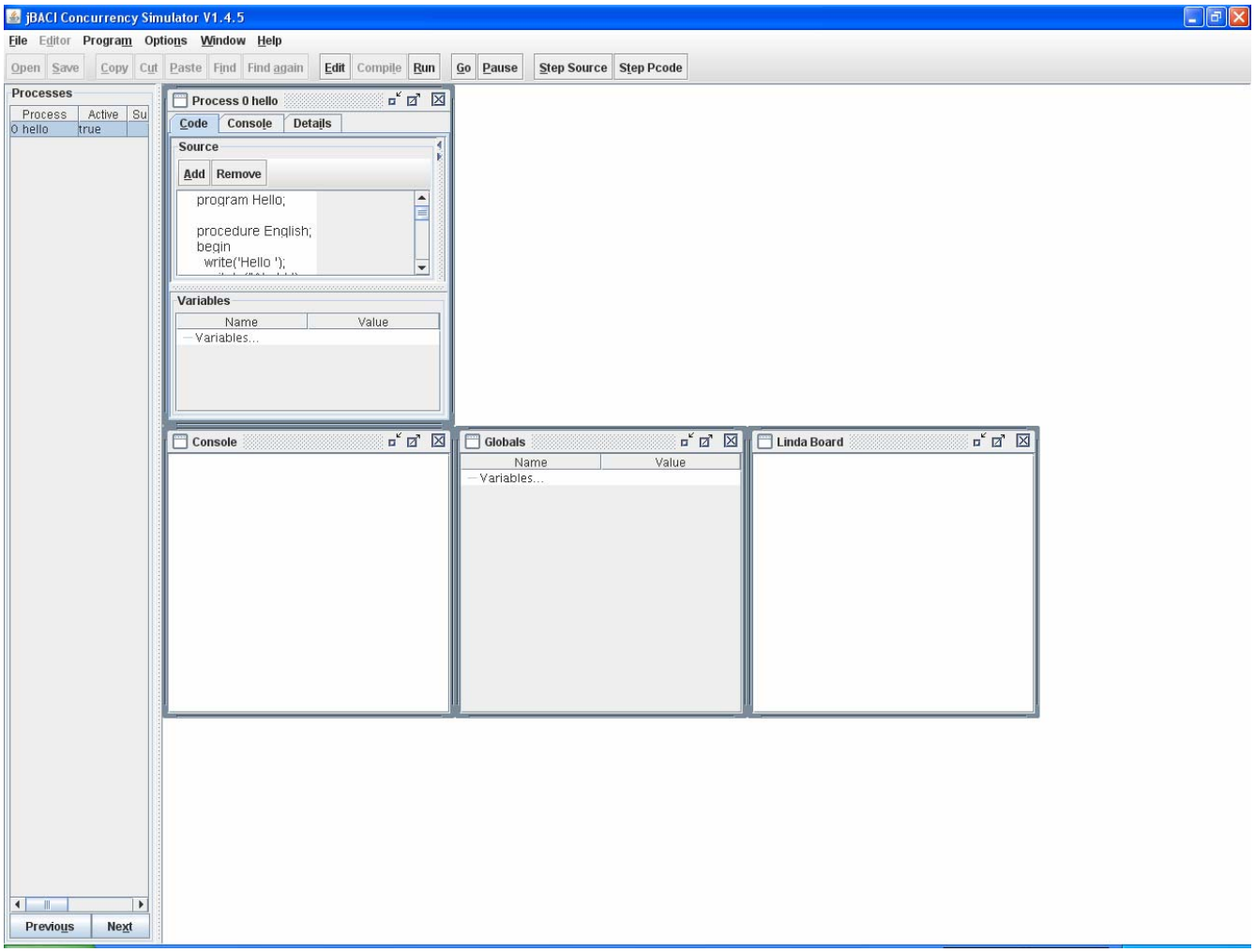
4. Käyttöesimerkki

Avaa lähdetiedosto valitsemalla Open File-valikosta. Tämä käyttöesimerkki pohjautuu jBACIn mukana tulevaa Hello.pm tiedostoon, joka löytyy examples-kansiosta. Tiedosto avautuu editointitilaan, jossa voit halutessasi muokata lähdekoodia ennen kääntämistä.



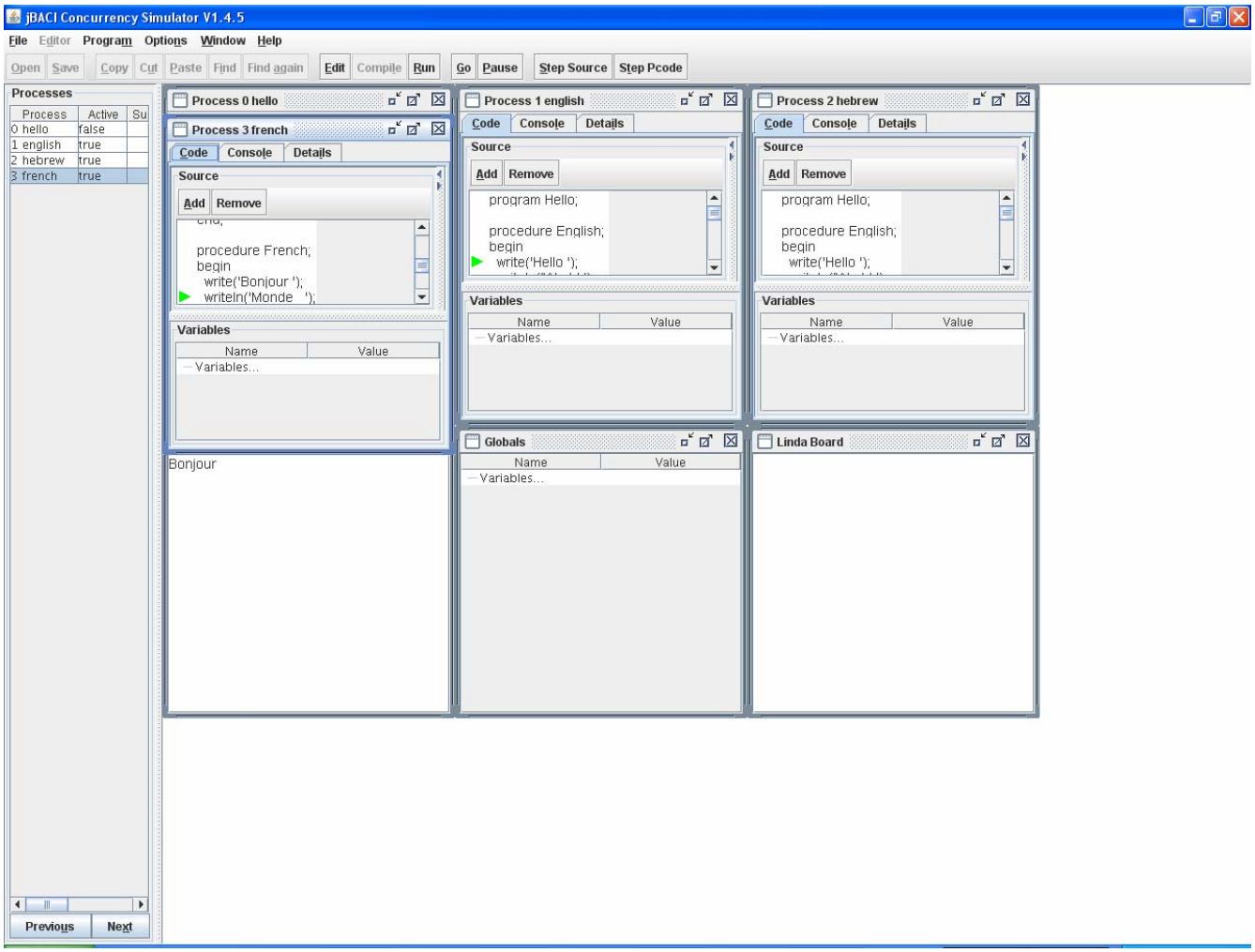
Kuva 4: Lähdekoodi editointitilassa

Käännä ohjelma painamalla Compile-painiketta. Ohjelma ilmoittaa käännöksen onnistumisen tai epäonnistumisen pop up –ikkunassa. Jos käännös onnistui, paina Run-painike, jolloin ohjelma siirtyy suoritustilaan.



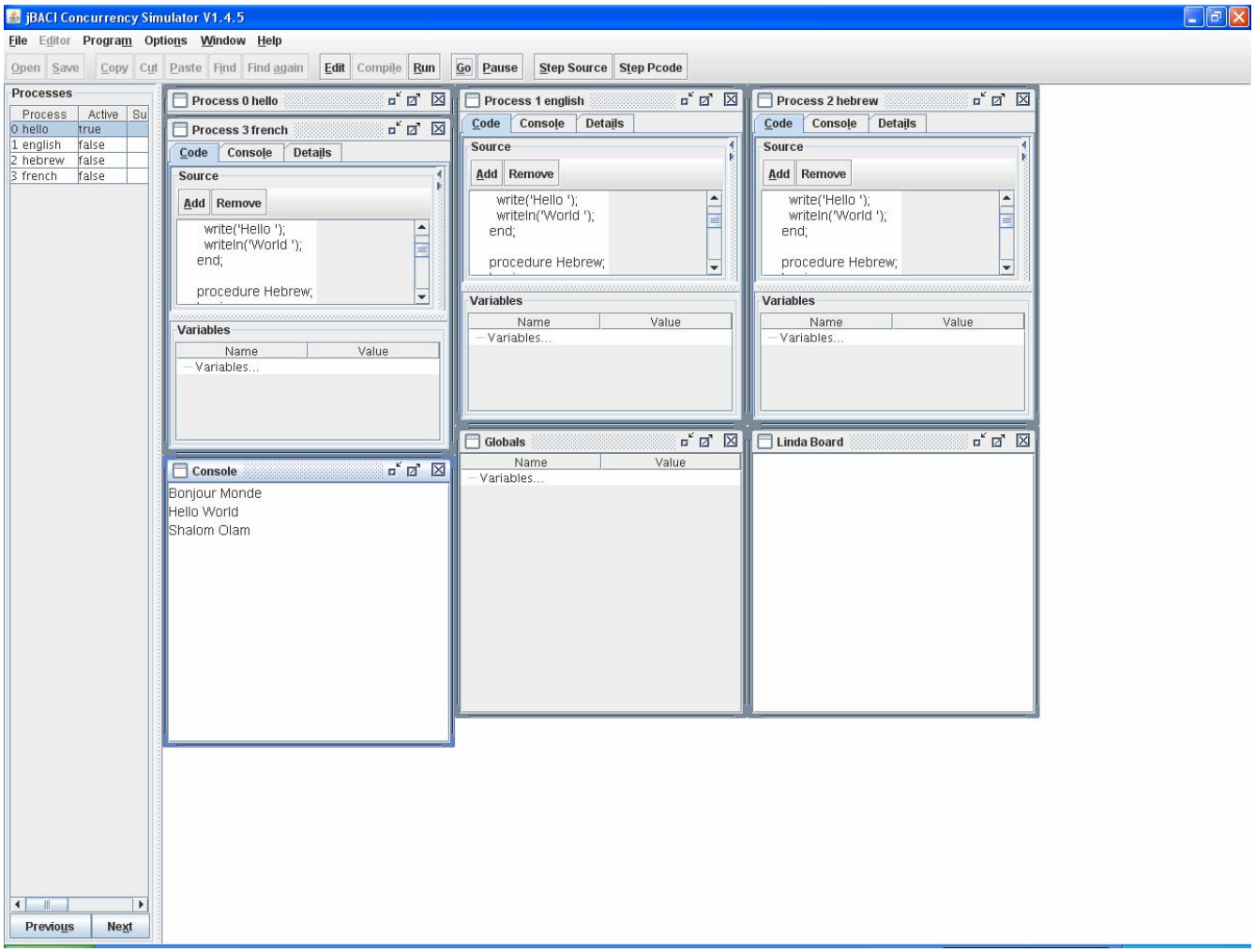
Kuva 5: Suoritustila ennen ajoa

Voit ajaa koko ohjelma kerrallaan painamalla Go-painiketta tai voit edetä vaihe kerrallaan painamalla Step Source –painiketta kuten alla on tehty. Jos kaikki haluamasi prosessit ei tule itsestään esille, voit valita niitä ikkunan vasemman reunan listasta.



Kuva 6: Ohjelmakoodia suoritetaan askel kerrallaan

Suorituksen aikana ohjelma tulostukset ilmestyy Console-ikkunaan ja globaalien muuttujien sen hetkiset arvot näkyvät Globals-ikkunassa.



Kuva 7: Ohjelmakoodi ajettu loppuun

5. Käyttöesimerkki 2

Rinnakkaisohjelmointi –kurssilla opiskelevan Tepon täytyy tehdä laskuharjoitustehtävänä koodinpätkä, jonka tulisi toimia sillä tavalla oikein, että koodi antaa tulokseksi ajettaessa aina saman vastauksen. Teppo päättää testata koodiaan jBACI –ohjelman avulla tietyn määrän kertoja, vaikka tietääkin, ettei se vielä itsessään riitä osoittamaan koodia oikeelliseksi. Hän kuitenkin haluaa jonkinlaisen suuntaa-antavan näkemyksen koodin oikeellisuudesta, ja uskoo olevan hyvin mahdollista, että koodi on oikein tehty jos jBACIlla 20 kertaa ajettuna saadaan tulokseksi aina sama vastaus.

-Teppo on jo asentanut jBACIn itselleen Windowsiin. Hän menee kansioon, jonne ohjelman tiedostot on tallentanut ja klikkaa ”run.bat”. Näin hän saa auki vielä tyhjän jBACIn ikkunan.

-Teppo valitsee jBACIn ylävalikosta ”file” ja sieltä ”open” ja hakee C -- -kielellä kirjoitetun ohjelmansa ”laskeyhteen.cm” tiedostosta, jonne sen on tallentanut. Koodi avautuu jBacin aiemmin tyhjään ruutuun.

-Teppo painaa toimintonapeista ”COMPILE” ja saa virheilmoituksen, jossa sanotaan: ”error near 'summa', line 11 of file (polku, jonne ’laskeyhteen.cm’ on tallennettu) Undeclared identifier 'summa’ at level 1 “. Tämän perusteella Teppo huomaa koodistaan sellaisen virheen, ettei hän ole muistanut lainkaan määritellä muuttujaa “summa” ennen kuin on käyttänyt sitä koodissa muuttujana.

-Teppo korjaa virheen ja määrittelee muuttujan summa. Uudelleen “compile” –nappia klikattuaan Teppo saa ilmoituksen: “Pcode and tables are stored in (polku\laskeyhteen.pco) Compilation listing is stored in (polku\laskeyhteen.pco)” Tästä Teppo päättelee, että kääntäminen on onnistunut ja painaa ”OK”.

-Tämän jälkeen Teppo painaa nappia ”RUN”, jolloin hänen eteensä ilmestyvät jBACIn pienet eri ikkunat.

-Teppo painaa nappia ”GO”, jolloin hänen ohjelmansa ajetaan. Teppo on tällä kertaa kiinnostunut vain ohjelman antamasta lopputuloksesta, joka on tällä kertaa 15, kuten oli odotettua. Teppo painaa uudelleen ”RUN” –nappia, tämän jälkeen ”GO”, ja sama toistuu. Tällä kertaa tulokseksi tuleekin 6, mikä ei ollut odotettua.

-Teppo palaa napista ”EDIT” tarkastelemaan koodiaan uudestaan. Aikansa pähkäiltyään Teppo huomaa käyttäneensä semaforia hieman virheellisesti. Teppo tekee tarvittavan korjauksen, suorittaa ohjelman kääntämisen uudelleen kuten äskenkin ja ajaa ohjelman. Tällä kertaa hän saa taas odotetun tuloksen.

-Teppo ajaa ohjelman yhteensä 20 kertaa ja saa aina saman odotetun tuloksen. Tästä Tepon epäilykset siitä, että koodi toimii oikein ja sen uskaltaa esittää laskuharjoituksissa vahvistuvat.

-Teppo sulkee ohjelman painamalla ruksia ohjelmaikkunan ylälaidassa.