Stichting NIOC en de NIOC kennisbank

Stichting NIOC (www.nioc.nl) stelt zich conform zijn statuten tot doel: het realiseren van congressen over informatica onderwijs en voorts al hetgeen met een en ander rechtstreeks of zijdelings verband houdt of daartoe bevorderlijk kan zijn, alles in de ruimste zin des woords.

De stichting NIOC neemt de archivering van de resultaten van de congressen voor zijn rekening. De website www.nioc.nl ontsluit onder "Eerdere congressen" de gearchiveerde websites van eerdere congressen. De vele afzonderlijke congresbijdragen zijn opgenomen in een kennisbank die via dezelfde website onder "NIOC kennisbank" ontsloten wordt.

Op dit moment bevat de NIOC kennisbank alle bijdragen, incl. die van het laatste congres (NIOC2023, gehouden op donderdag 30 maart 2023 jl. en georganiseerd door NHL Stenden Hogeschool). Bij elkaar bijna 1500 bijdragen!

We roepen je op, na het lezen van het document dat door jou is gedownload, de auteur(s) feedback te geven. Dit kan door je te registreren als gebruiker van de NIOC kennisbank. Na registratie krijg je bericht hoe in te loggen op de NIOC kennisbank.

Het eerstvolgende NIOC vindt plaats op donderdag 27 maart 2025 in Zwolle en wordt dan georganiseerd door Hogeschool Windesheim. Kijk op www.nioc2025.nl voor meer informatie.

Wil je op de hoogte blijven van de ontwikkeling rond Stichting NIOC en de NIOC kennisbank, schrijf je dan in op de nieuwsbrief via

www.nioc.nl/nioc-kennisbank/aanmelden nieuwsbrief

Reacties over de NIOC kennisbank en de inhoud daarvan kun je richten aan de beheerder:

R. Smedinga kennisbank@nioc.nl.

Vermeld bij reacties jouw naam en telefoonnummer voor nader contact.

Hulpmiddelen in een Programmeeromgeving

P. Vanneste, K. Bertels, B. De Decker
Katholieke Universiteit Leuven

Inleiding

Binnen het programmeeronderwijs besteden studenten een niet te verwaarlozen tijd aan schrijven van programma's. Veelal worden deze programma's ook ingetikt en uitgetest op een computer. De gebruikte programmeeromgeving kan een aantal hulpmiddelen bevatten die dit proces ondersteunen en vergemakkelijken. Deze presentatie geeft een overzicht van de verschillende mogelijkheden op dit gebied, en besluit met een bespreking van een ideale, geïntegreerde programmeeromgeving. We gaan er in deze tekst van uit dat er gewerkt wordt met een Pascal-achtige taal.

Hulpmiddelen bij het programmeren

Hulp bij het opbouwen en het intikken van het programma

- Syntax-gedreven editors
  Een syntax-gedreven editor ondersteunt de student tijdens het intikken van zijn programma. Een dergelijke editor laat de student enkel toe om syntactisch correcte programma's in te tikken. Daar waar de syntax bepaalde keuzes laat, kan de student één daarvan selecteren uit een lijst. Indien een bepaalde taalconstructie (if-then-else, while, ...) gekozen werd, dan wordt deze op het scherm gebracht. De student kan enkel nog de variabele gedeelten van die constructie invullen. We bespreken enkele voorbeelden van syntax-gedreven editors en hun belangrijkste voor- en nadelen.

- Kennis-gedreven editors
  Een kennis-gedreven editor vertrekt van de veronderstelling dat een programma opgebouwd wordt met behulp van hoog-niveau plannen. Een plan kan bestaan uit een aantal subplannen of, op het laagste niveau, programma-code. Om een programma op te bouwen selecteert de student een aantal plannen die hij nodig acht om het gestelde probleem op te lossen. Deze plannen kunnen dan met behulp van de kennis-gedreven editor verder verfijnd worden. Hierbij neemt de editor zoveel mogelijk werk voor zijn rekening. Op het code-niveau komt een kennis-gedreven editor overeen met een syntax-gedreven editor.

Hulp bij het uittesten en het verbeteren van fouten

De omgeving kan ook hulp bieden voor het nagaan van de correctheid en het lokaliseren van eventuele fouten.

- Debuggers
  Een debugger ondersteunt de student bij het opsporen en lokaliseren van eventuele fouten. We maken een onderscheid tussen een interactieve en een niet-interactieve debugger.
  * Bij een niet-interactieve debugger wordt het programma eerst volledig uitgevoerd. Daarna krijgt de student informatie over het uitgevoerde programma: hoeveel keer werd elke instructie uitgevoerd, wat is de eerste en laatste waarde van variabelen, ...
  * Een interactieve debugger laat de student toe om het programma instructie per instructie uit te voeren en op elk moment tijdens de uitvoering de waarde van een variabele te onderzoeken (en eventueel te wijzigen).

  Het voordeel van een dergelijke omgeving ligt vooral in het uitwinnen van tijd die nuttig besteed kan worden voor meer oefeningen.

- Een grafische debugger
  Indien we de variabelen en hun waarden grafisch voorstellen, bekomen we een omgeving voor de visualisatie van programma's. Zo'n omgeving is handig omdat ze toelaat variabelen op een conceptueel niveau te bekijken (een lineaire lijst wordt getekend, zodat de informatie veel eenvoudiger te interpreteren valt dan een reeks getallen die dezelfde lijst voorstelt). Een dergelijke omgeving kan ook nuttig zijn tijdens het opbouwen van programma's en draagt bij tot een beter begrijpen ervan. We bespreken kort het verschil met animatie-omgevingen die visualisatie gebruiken om het effect van algoritmen te illustreren.

- Interpretatie van asserties
  Het is belangrijk dat een student ook op een formele manier kan nadenken over de correctheid van zijn programma's. Een programmeeromgeving kan dit ondersteunen door de student toe te laten om binnen zijn programma's asserties op te nemen die de relaties tussen verschillende variabelen op een bepaald moment van de uitvoering beschrijven. Tijdens de uitvoering wordt dan nagegaan of aan die assertie voldaan is.

Automatische beoordeling van programma's
Hier moet de student niet zelf op zoek gaan naar eventuele fouten. Zijn programma wordt automatisch beoordeeld, en de student krijgt een overzicht van de aanwezige fouten. Om een programma te beoordelen moet er op één of andere manier een beschrijving van de opgave opgeslagen zijn. De meest eenvoudige hulpmiddelen voeren het programma uit met een aantal testvoorbeelden en vergelijken de uitvoer met de verwachte uitvoer. Andere hulpmiddelen voeren een diepgaande analyse van het programma uit.

Integratie
Er zijn veel verschillende programmeeromgevingen die elk één van de opgesomde hulpmiddelen bevatten. We bespreken hoe een volledig geïntegreerde omgeving er zou kunnen uitzien.

Slot
Deze presentatie geeft een overzicht van bestaande hulpmiddelen ter ondersteuning van het programmeerproces tijdens het leren programmeren. De verschillende hulpmiddelen worden besproken, en er wordt gepleit voor het bouwen van een programmeeromgeving die de verschillende hulpmiddelen integreert.