**Функционално програмиране**

***Кратък предговор***

Dr Scheme е интерактивна, интегрирана среда за програмиране, създадена специално за нуждите на начинаещите програмисти.

Преди да използвате Dr Scheme трябва да изберете ниво на езика.В противен случай, когато се опитате да стартирате въведена програма, ще се изобрази екран за грешка, в който трябва да изберете съответния тип език за програмиране. Този тип на и ниво на езика винаги може да бъде променено като се използва менюто Language -> Choose language.

1. ***Въведение в Scheme***

Тук ще въведем някои основни понятия за начинаещите програмисти, които никога досега не са се занимавали с функционално програмиране и не са се срещали с езика Scheme. Представени са два малки примера, които могат да илюстрират основните операции и методи на работа в Scheme.

Обичайната първа програма е тази, която разпечатва "Hello, World!" на конзолата. Използвайки редактора на Scheme, създаваме файл наречен hello.scm и въвеждаме следния код:

***;The first program***

***(begin***

***(display "Hello, World!")***

***(newline))***

Първия ред е коментар. Когато Scheme види точка и запетая, той ги игнорира, както и всичко останало до края на реда.

Begin - формата в Scheme е начин за въвеждане на последователност от поддекларации. В този случай има две поддекларации. Първата е извикване на разпечатващата процедура, която извежда аргумента си (низа "Hello, World!") на конзолата или стандартния изход. Това е последвано от извикване на друга процедура, която извежда нов ред.

За да изпълнети тази програма, първо трябва да стартирате Scheme. След това се въвежда кода в реактора и се стартира с Run. Scheme ще изпълни съдържанието на файла hello.scm и ще разпечата низа "Hello, World! ".

Нека да разгледаме следната проста математическа функкция:

f2c (t) = 5/9 x (t - 32)

Тази функция преобразува градусите по Фаренхайд в градуси по Целзий. Сравнително лесно е да се конвертира тази функция в програма на Scheme: използваме ***define*** за да въведем нова дефиниция и поставяме оператора преди операндите.

(**define** (f2c t) (\* 5/9 (- t 32)))

Тази функция трябва да се въведе в горната половина на прозореца на Dr Scheme и след това се стартира с бутона Run. Това зарежда функцията в долния прозарец, който е един своеобразен интерактивен калкулатор. За да го тествате може да въведете три стойности: -40, 32, 212. Това може да стане чрез следните условия:

f2c(-40)

f2c(32)

f2c(212)

В Scheme това трябва да бъде записано по следния начин:

(f2c -40)

(f2c 32)

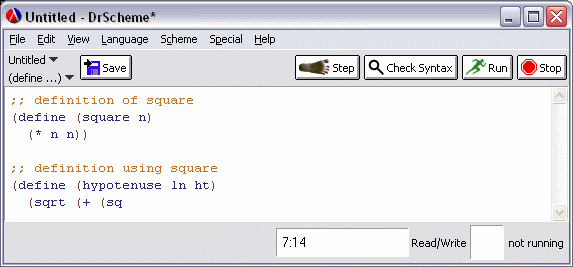
(f2c 212)

Опитайте и вижте какви са резултатите, които се получават! Защо Scheme извежда тези стойности? Тук е мястото където може да използвате т.нар. ***степер***. Той ще ви покаже стъпка по стъпка резултатите, които получава от изразите. За да го използвате, сложете някакъв израз, който трябва да бъде пресметнат (напирмер (f2c -40)) в горната част на прозореца, след което стартирайте с бутона Step.

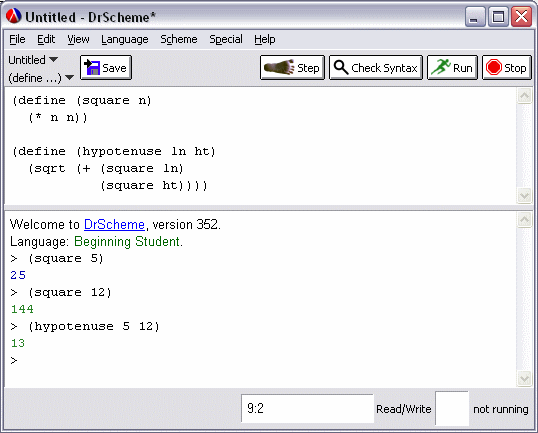
1. ***Интерактивна среда***

***Въведение***

Dr Scheme предоставя интерактивна среда. Горния, или дефиниращия прозорец, е пълнофункционален графичен редактор. Редаторът поддържа синтактично оцветяване на кода, което съответства на правилата на синтаксиса на Scheme (и следователно е точно, по начин по който много други оцветяващи системи за синтаксис не са).



Долния, или интерактивния прозорец, имплементира в себе си цикъл четене - изпълнение - извеждане на резултат (read - eval - print loop (REPL)).



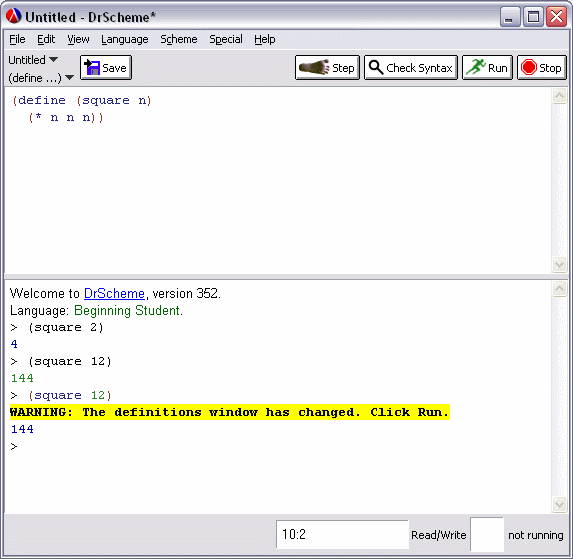
Бутонът Run зарежда описанието на кода от горния (дефиниращ) прозорец в долния (интерактивен прозорец).

***Прозрачният REPL***

Дистрибуцията на REPL, включена в Scheme е т.нар. прозрачен REPL. Така се избягват много типични грешки и недоразумения. REPL в Dr Scheme избягва тези грешки по два начина:

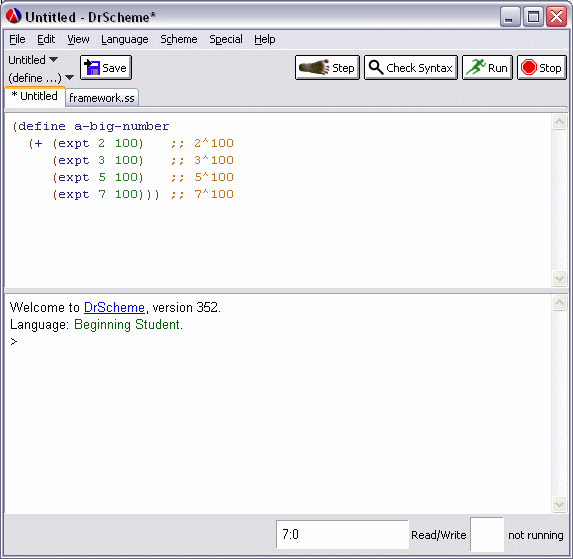
- Всеки път, когато потребителят кликне Run, Dr Scheme изтрива всички вътрешни състояния, свързани с REPL.

- Dr Scheme предупреждава потребителите, когато те правят променир да не стартират изпълнение на програмата с Run и да прехвърлят първо новите въведени неща в горния прозорец в REPL.



***Въвеждане в раздели***

Редактора на Dr Scheme позволява да редактирате множество файлове в отделни раздели, а не да се налага да редактирате всеки файл в свой собствен прозорец. Менюто File - > New Tab създава нов раздел, а File -> Close Tab затваря определен раздел (ако има повече от един).



1. ***Нива на езика за програмиране Scheme***

***Въведение***

Начинаещите правят грешки и се нуждаят от обратна връзка, която да е съобразена с тяхното ниво на разбиране на езика. Синтаксисът на Scheme е толкова толерантен, че повечето Scheme - реализации са лоши при отчитането на грешките в полезни изрази, или терми както се наричат в терминологията на Scheme.

Scheme преодолява тези проблеми чрез осигуряване на йерархия на езиковите нива. Йерархията съответства на естествено въведените в Scheme нива:

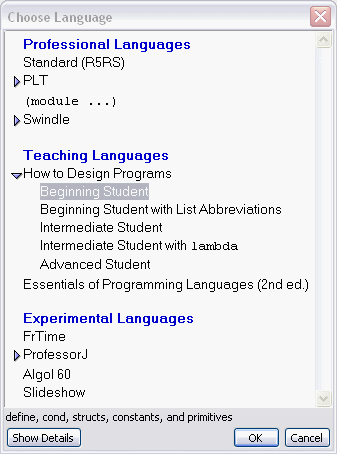
- Език за начинаещ студент (beginning student language), който включвафункции, структури и списъци.

- Език за средно-напреднал студент (intermediate student language), в който са добавени обхватни лексикални конструкции.

- Eзик за напреднал студент (advanced student language), в който се позволява мутация и поддържа удължавания.

Нивото на Scheme езика всъщност е стандартен Scheme с или без графики и поддръжка на система за отстраняване на грешки (debugging).

Тези нива налагат други синтактични и семантични ограничения, основани на обичайните грешки, които се наблюдава, че студентите правят през годините. Всяко ниво се опитва да докладва грешки, съответни на това ниво.



Когато стартирате Scheme ще бъдете насочени първо да изберете съответно ниво на езика.

***Примери***

Да вземем за пример един начинаещ студент, който е свикнал на инфиксния запис функциите: f (x). Този студент може случайно да се върне към този тип запис, програмирайки на Scheme, например в клаузата cond.

(**define** (length l)

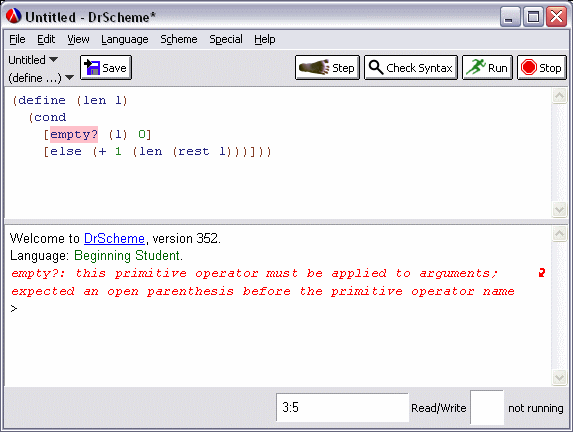
(**cond**

[empty? (l) 0]

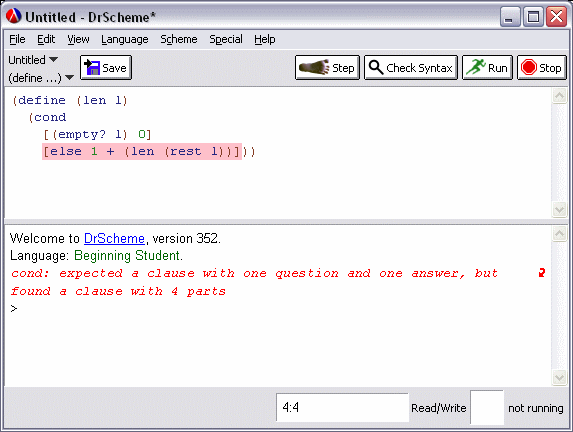
[**else** (+ 1 (length (rest l)))]))

Тази дефиниция е синтактично правилна в конвенционалния Scheme и традиционното изпълнение ще даде сигнал за грешка само при изпълнение на функцията. Тъй като начинаещия не знае за функциите от първи ред, съобщението за грешка за него ще бъде неразбираемо.

Scheme ограничава синтаксиса на езика на начинаещи до прост език от първи ред. Чрез ограничаването на езика Scheme може да отчете типични грешки на начинаещи, като тази описана по - горе. В този пример се подчертава също и самата клауза - нарушител, което предполага по - естествена редакция от страна на начинаещия. В този пример, Scheme улавя използването на примитивна функция без аргументи.



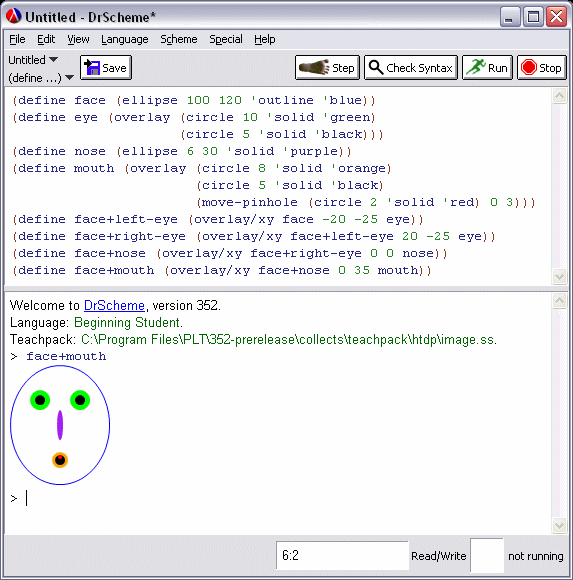
В следващия пример Scheme улавя пропуск в инфиксната нотация, като забелязва че условието има твърде много подизрази.



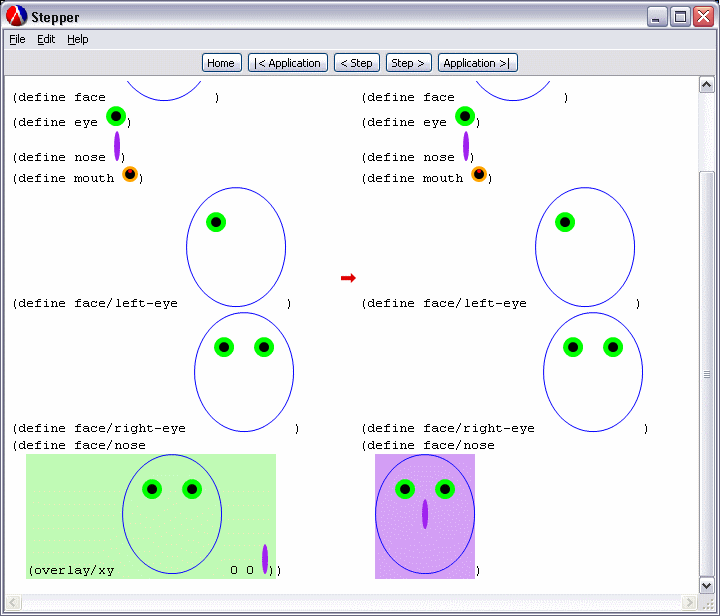
1. ***Изображенията като стойности***

***Създаване на изображения с код***

Scheme поддържа използване на изображения като обикновенни стандартни Scheme - стойности. Следващата програма демонстрира конструирането на лице чрез последователност от операции.

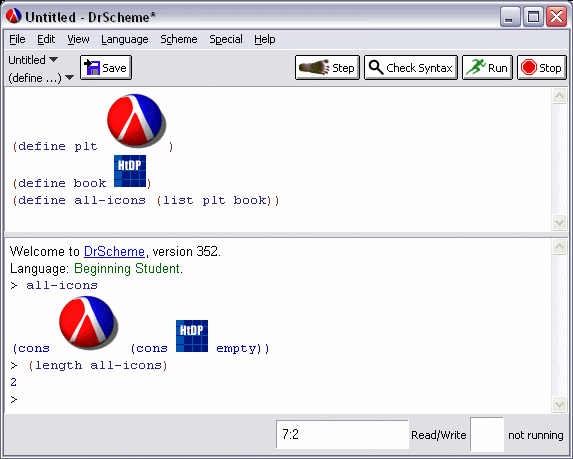


Степера (който беше описан и по - рано) поддържа изображенията като стойности, така че програмистът може да проследи и да види горната програма в действие.



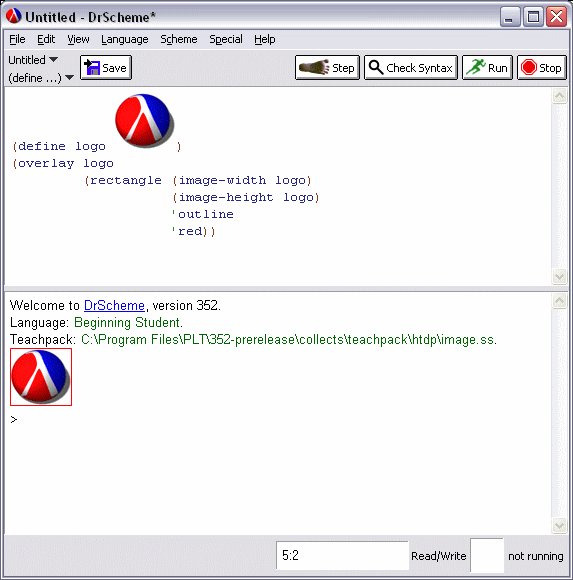
***Включване на графични фйлове в source- кода***

Програмния source - файл също може да съдържа изображения. Scheme съхранява такива файлове в обичаен, многоплатформен, преносим формат. Изображенията са стойности от първи ред, което означава че програмистите могат да ги подават към функции и да ги получават като върната стойност на функция, да ги съхраняват в структури от данни и т.н. Използвайте Special -> Insert Image, за да вмъкнете изображение в редактирания файл в горния, дефиниращ прозорец.



***Редактиране на графични файлове***

Естествено, програмите могат да включват външни графични файлове и да те бъдат редактирани чрез source - кода.



1. ***Спецификация и изобразяване на числа***

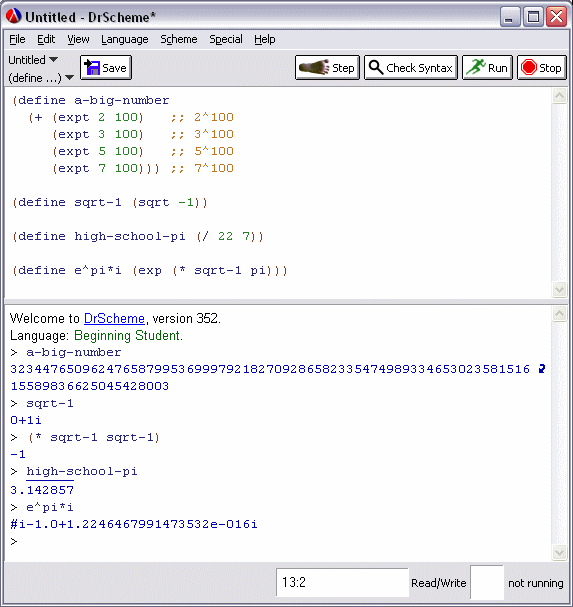
Scheme предлага изключителна поддръжка на числа. Следващият скрийншот илюстрира няколко от многото характеристики, които scheme притежава:

- големи числа (т.е. такива, които са по -големи от естествената хардуерна поддръжка на машината)

- точно разпечатване на рационални числа (различните нива на езика ги печатат по различен начин)

- изрично сигнализиране за неточни числа (обозначението **#i**)

- комплексни числа



1. ***Пакети за обучение***

***Въведение***

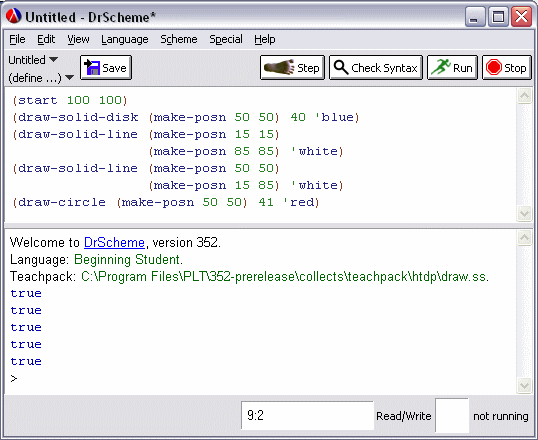
Учителите и професорите често свързват студентския код с библиотеки, включващи напреднали практики за графика, работа в мрежи и т.н. Тези библиотеки не могат да бъдат заредени с помоща на просто въвеждане на команди, тъй като те могат да използват функции, които не присъстват в езиковото ниво на ученика.

Scheme решава този проблем с помоща на пакети за обучение. Те могат да използват пълния език, зареждат се само веднъж при стартирането на компилатора на Dr Scheme и автоматично се извикват всеки път, когато потребителят натисне Run. Scheme идва с няколко пакета за обучение, които изпълняват библиотеки, като например такива поддържащи игра и включващи различни кодове за игри. Езиковите нива и пакетите за обучение дават най - доброто от два отделни свята: показване на адекватни типове грешки за начинаещи и разширени функции за създаване на интересни уроци и демонстриране на приложения.

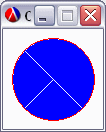
Пакетите за обучение се засягат във все повече текстове и уроци за студенти.

***Пакети за обучение за графика***

Един доста често използван пакет за обучение е библиотеката за графика за начинаещи. Тя осигурява опростен интерфейс към обширния GUI (Graphic User Interface - графичен потребителски интерфейс), достъпен в Scheme (който всъщност е използван и за изграждането и на самата среда Scheme).



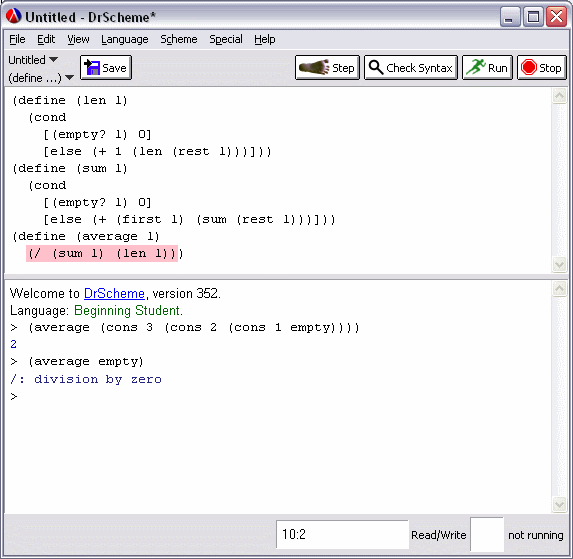
Изходът от тази програма е следното изображение:

******

1. ***Сигнализиране за грешки по време на изпълнение***

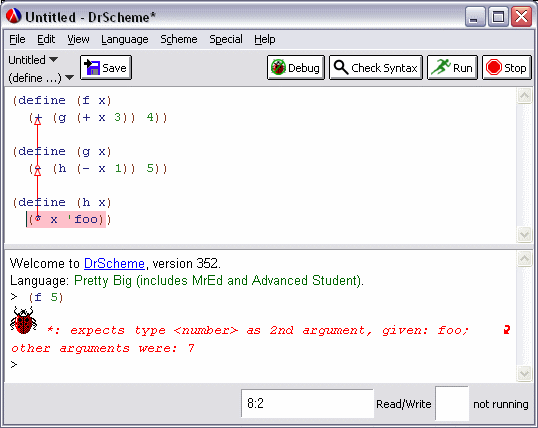
***Отбелязване и подчертаване на source - кода***

Scheme подчертава местоположението не само на синтактичните, но и на динамичните грешки, както илюстрира и следния пример:

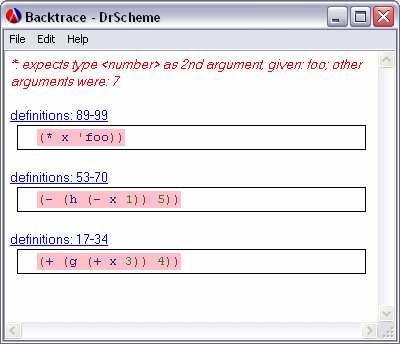


***Обратно проследяване***

В нивата на стандартния и PLT Scheme - език, когато дадена програма съдържа динамична грешка, Scheme подсказва това на потребителя със съобщение за грешка и с икона за грешка. Той също така наслагва и червени стрелки към източника, за да се подчертае управляващата последователност, която води до грешката.



Кликайки на иконата за грешка (която представлява всъщност калинка - от англ. Bug - буболечка), се извиква прозореца за обратно проследяване, който показва последователността от извиквания, водещи до грешката (Това съответства на извикванията, които стрелките показват). Кадрите са придружени и от хипервръзки, насочващи към кода - източник на грешката.

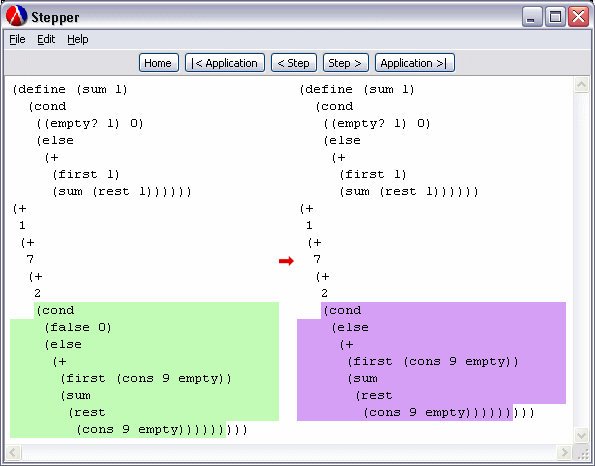


1. ***Постъпково изпълнение и отстраняване на грешки***

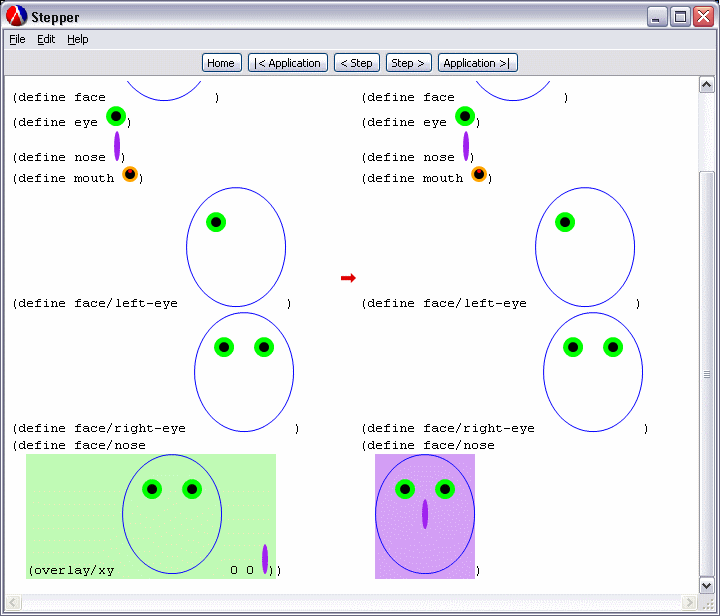
Scheme предлага инструменти за постъпково изпълнение и отстраняване на грешки. **Степерът** предлага алгебричен вид на изпълнението на програмата, подходящ за студенти, а **дебъгерът** е по - близо до компилиращ инструмент в програмна среда за професионалисти.

***Степер***

Степера позволява на потребителите да изпълняват постъпково всяка редукция на програмата. Той използва оцветяване, за да подчертае попълненията и редукциите в програмата. Потребителите могат да се придвижват стъпка напред или назад през историята на модификацията и редукцията на програмата. Това илюстрира алгебричния характер на програмиране.



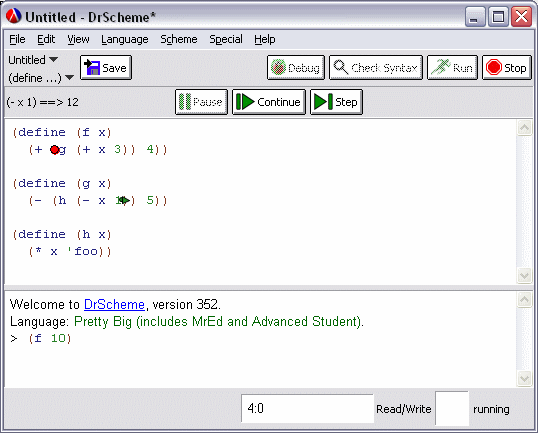
Степерът може да се справи дори с програми, които използват изображения като стойности.



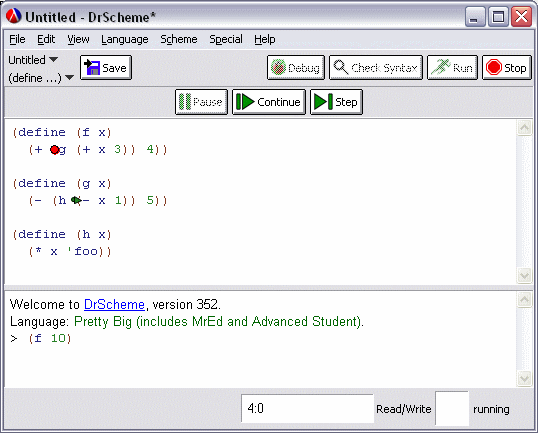
***Дебъгер***

Когато използваме език, който включва всичко от Scheme, като Standard или Pretty Big, бутона на степерът се променя и се означава като дебъгер. Може да стартирате програми с или без активиран дебъгер. Когато използваме Run, програмата се изпълнява по обичайния начин. Използването на Debug е като използването на Run, с изключение на това, че се включва и информация за корекции и отстраняване на грешки. Дебъгера включва три бутона: Pause, Continue, Step.

Бутона Step е всъщност изпълнение на следващата функция. На всяка стъпка Scheme показва изразът, който предстои да бъде изпълнен със зелена стрелка.



По време на изпълнението може да вмъкнете точка на прекъсване. Задръжте мишката върху израза и ще се появи червено оцветяване.



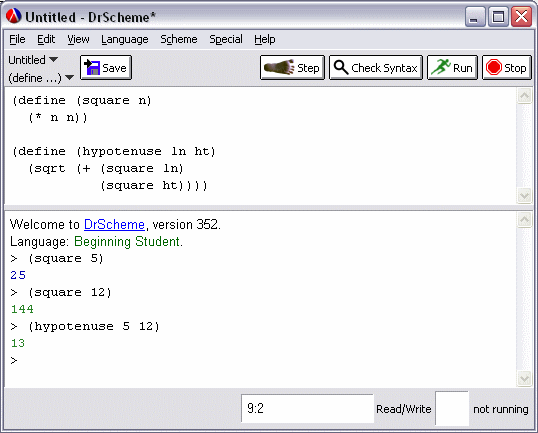
При кликане с десния бутон (или на Macintosh - при кликане на Control) върху точката се появява меню, което ви позволява да добавите или да премахнете точка на прекъсване. Бутона Continue продължава изпълнението до завършване на програмата, до срещане на точка на прекъсване или спира със съобщение за грешка.

Бутона Pause дава възможност на потребителя да върне контрола на дебъгера по време на една продължителна серия от изчисления (като loop - цикъл). При щракане с дения бутон (или Control - бутон) върху идентификатора, ако е в рамките на текущото изпълнение, се показват стойността на идентификатора и се дава възможност за промяна на стойността му.

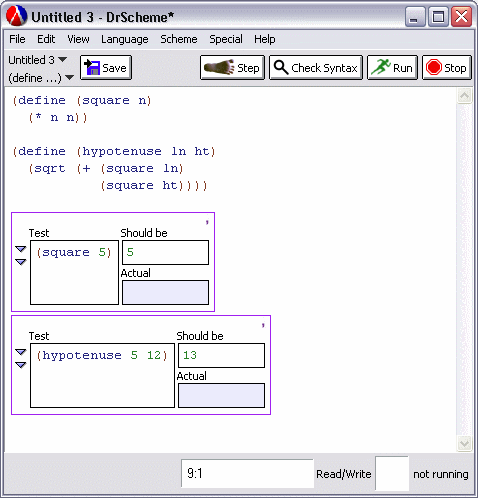
1. ***Тестване***

***Включване на тестове в кода на програмата***

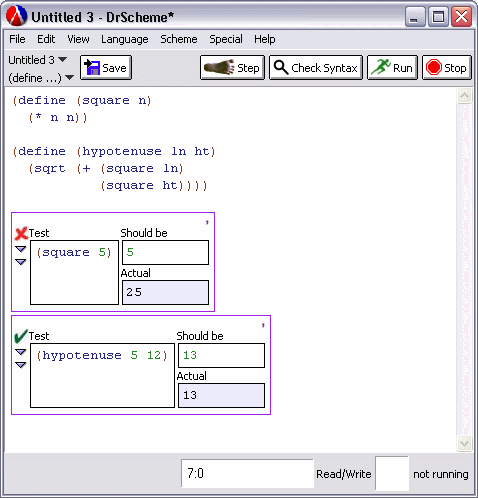
Scheme прави относително лесно изчистването и капсулирането натестови примери. да разгледаме за пример програмата, която разгледахме и преди.



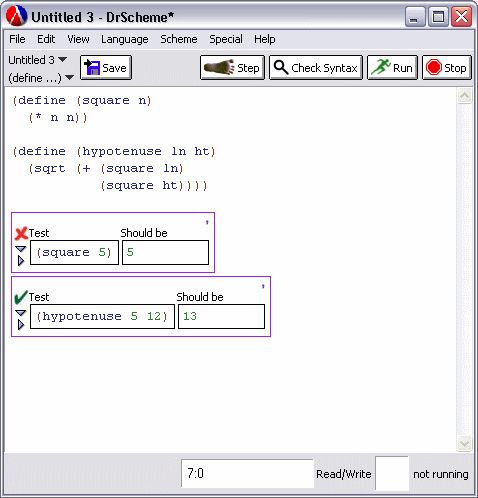
Тъй като REPL се разчиства на всяко ново стартиране чрез Run, става доста досадно да бъдат въвеждани многократно тестовете в REPL. Вместо това програмистът може да вгради тестовете в source - кода на програмата с помоща на Special -> Insert test cases. Всеки тест може да има самостоятелен етикет, последван от израза на теста и очаквания отговор. Тези последни две кутии могат да съдържат произволен scheme - код.



Кликвайки Run се обозначава коректността на всеки тест.

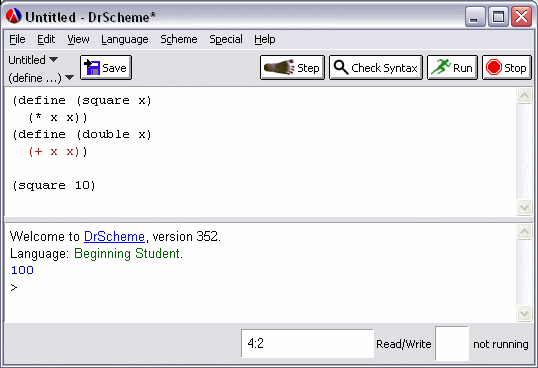


Пълния тест може да заеме голяма част от мястото на екрана, но програмиста винаги може да свие теста като кликне върху триъгълника, оставяйки само описатения етикет и индикацията дали теста е бил коректен или се е провалил.

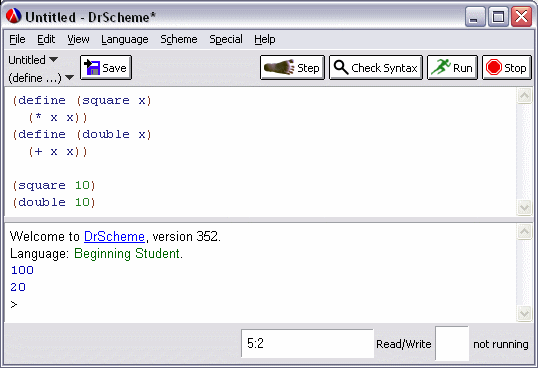


***Обхват на кода***

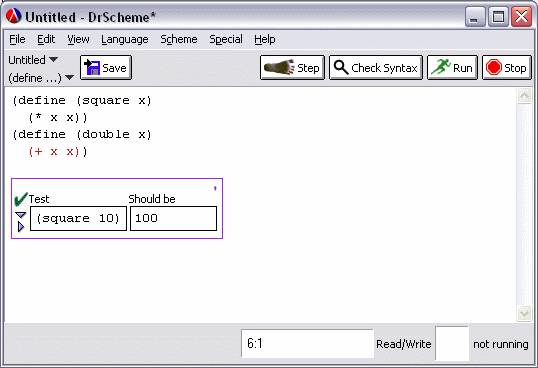
В нивата на езика за студенти Scheme автоматично показва обхвата на изразите от кода на програмата. Това е една добра основа за студентите и преподавателите, с която може да се определи дали дадена програма е била тествана по подходящ начин. Когато изразът не е бил изпълнен scheme го подчертава в червено, оставяйки останалата част от кода в черно:



Ако обаче всички изрази са били тествани, редактора връща целия код на програмата оцветен:



Представянето на обхвата разглежда тестове в областта на вградените тестови примери. По този начин, ако един тест изпълнява само функция square, Scheme представя програмния код в черно, подчертавайки неизпълнения код на функция double



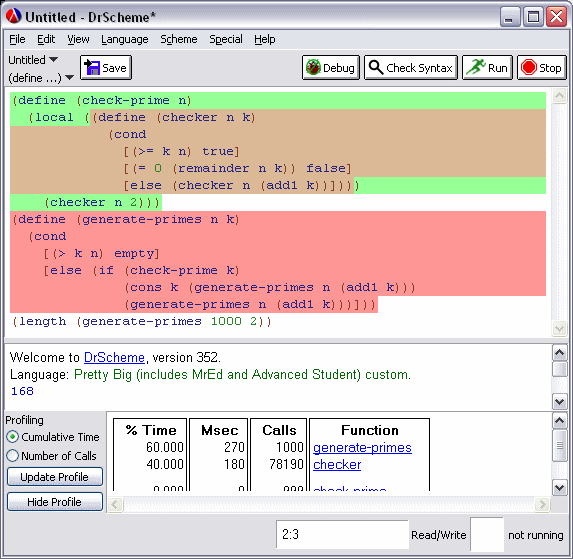
1. ***Профайлър***

Scheme съдържа профайлър за изпълнение на различните нива на процедурите. Профайлъра събира информация по време на изпълнението на програмата, която показва графично. Профайлъра може да подреди изхода в съответствие с броя на извикванията или с времето за изпълнение. За да активирате профайлъра:

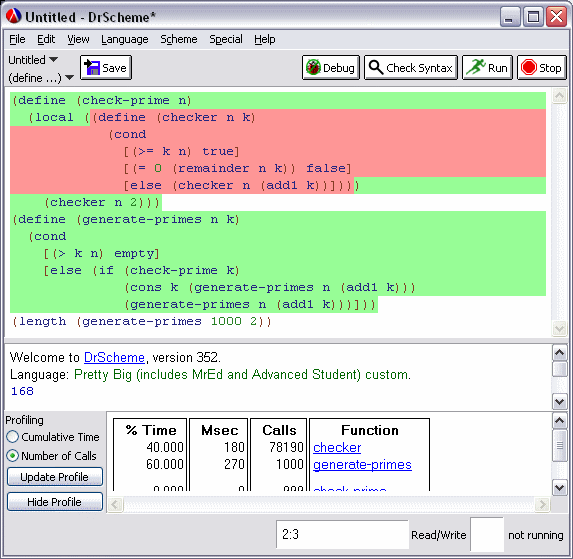
* Трябва да инструктирате Scheme да генерира профайлърна информация. За да направите това трябва да изберете **Show details** в менюто **Language** -> **Choose Language** и да изберете профилиране в панела **Dynamic Properties** (не е нужно да изпълнявате тази стъпка, ако промените конфигурацията на езика).
* Стартирайте програмата
* Изберете **View** -> **Show Profile**.

Профайлърът включва както панел, така и оцветяване на кода на програмата: топлите цветове показват по - голямо използване, по - студените цветове показват по - малко използване. За да промените цветовете, използвайте раздела Profiling в Edit -> Preferences.

Двете изображение по - долу са от едно и също изпълнение на една и съща програма, която генерира списък с прости числа. Първото изображение показва резултата от избора на профилиране по време (time profiling), което е по подразбиране.



Второто изображение показва резултата от пофилиране по броя на извикванията:

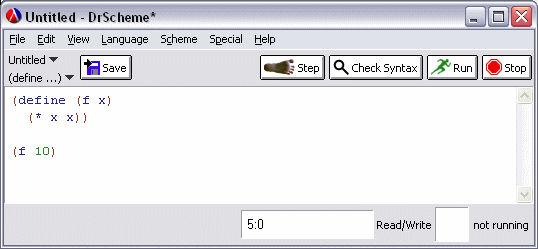


1. ***Програмно разглеждане***

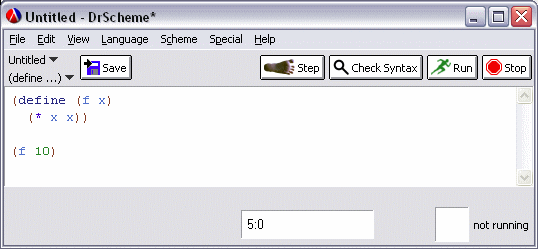
Scheme предлага няколко начина за програмно разглеждане

***Проверка на синтаксиса***

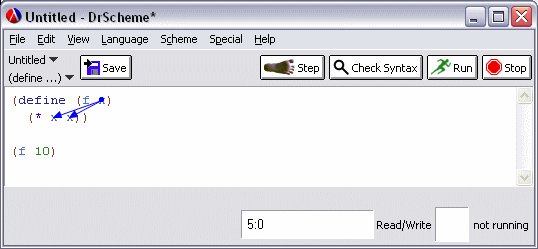
Всеки дефиниращ прозорец има бутон **Check Syntax**. При натискането на този бутон, Scheme анализира програмния код и идентифицира синтактичните грешки, без да се налага да стартирате програмата. Той също така помага за намирането на обвързани и ограничени екземпляри на идентификаторите. Да предположим, че запчнем с тази програма:



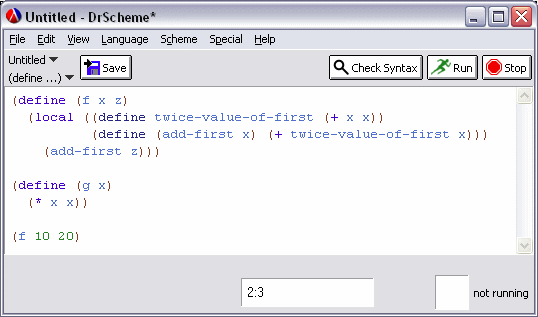
Кликването на Check Syntax предизвиква оцветяване на идентификаторите и те стават по - бледи.



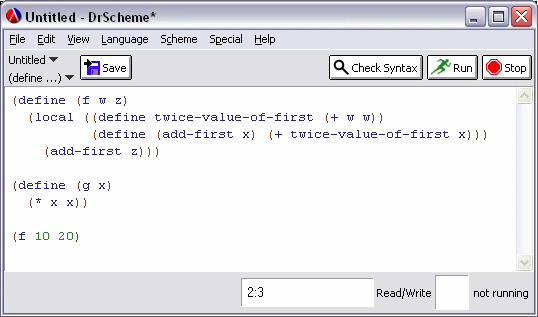
При движение с мишката над свързаните идентификатори се оказва тяхното свързване, а при задържане на курсора на мишката над ограничен идентификатор се показват всички негови ограничени използвания. При кликане с десния бутон на мишката се показва меню, което включва командата **Tack| Untack Arrow(s),** чрез която на екрана се показват стрелки, докато скролирате.



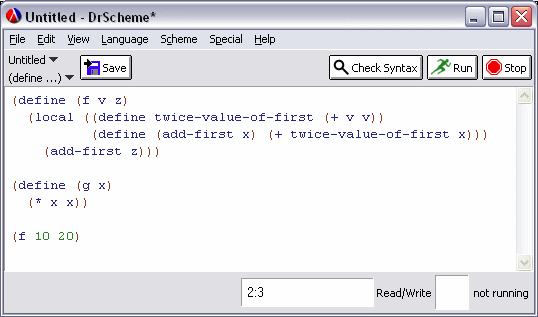
В същото меню е вслючена и команда за последователно преименуване на идентификаторите. Scheme използва "умен" механизъм за преименуване, който може да прави разлика между различните идентификатори, които имат едно и също име. За да видите как работи това, предлагаме да започнем с тази програма, след като вече сме кликнали Check Syntax.



При кликане с десния бутон на мишката върху **х** , която е ограничена във функцията **f**, се появява меню, което включва "**Rename x**". При избора на тази команда и въвеждане на w в полето, резултата е следната програма:



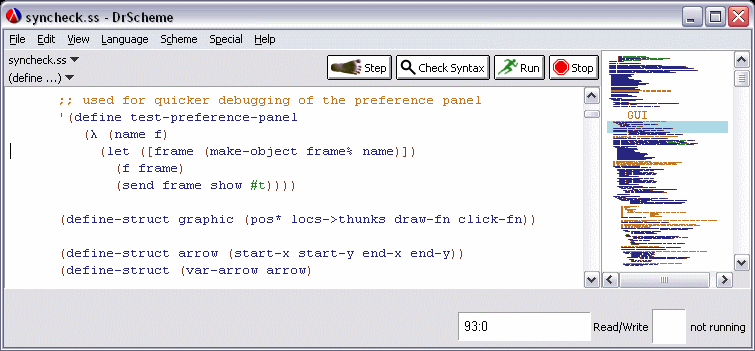
Забележете, че нито един от идентификаторите х във функцията **add-first** и **g** не са засегнати от това преименуване. Ако изберем първото **w** от добавените и го преименуваме на **v**, това автоматично ще промени и второто w в сумата.



Имайте предвид, че всяко преименуване или редактиране потенциално обезсилва информацията получена от проверката на синтаксиса чрез **Check Syntax**, така че е необходимо да проверяваме синтаксиса с Check Syntax след всяко преименуване на идентификатори.

***Контури***

Контура е изключително смалена версия на кода на програмата. Може да мислим за него и като "гледка от разстояние 20 000 метра". Менюто Special - > Insert Large Letters вмъква коментар с "големи букви", който помага за подчертаване на частите на кода.



***Дефиниции***

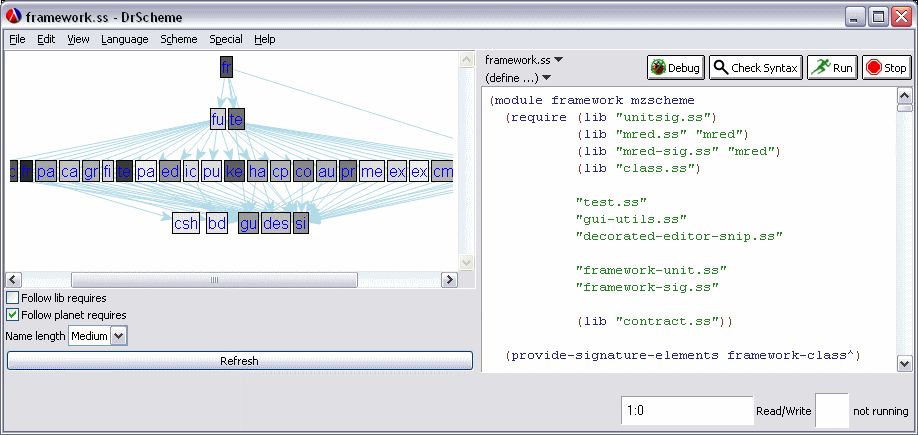
В горния ляв ъгъл на Scheme има едно поле с надпис

(**define** ...)

кликайки на това поле се появява един прозорец, в който се изписват всички дефиниции в кода на програмата.

***Модулно разглеждане***

В PLT езика, ако сте изградили програма с помоща на няколко модула (езикова конструкция в PLT Scheme), може да приложете командата **Module Browser**, достъпна от менюто Show.



Всяка кутия представлява модул, а стрелките представляват връзки и зависимости. При поставяне на курсора на мишката на някоя кутия се подчертават нейните връзки и се показват нейното име и размер (кутиите също така са потъмнени, пропорционално на размерите на съотетните модули). Двукратното щракване на кутиите е един добър и лесен начин да се придвижвате в кода на програмата. Редактора е полезен също и при откриване и премахване на ненужни препратки към модула.

Ако искате **Module Browser** като отделен прозорец, а не като панел във вашия редактор, може да изберете **Scheme** -> **Module Browser**.

1. ***XML кутии***

Scheme предоставя поддръжка за създаване на XML изход (който като частен случай може да се използва и за генериране на HTML). Това се съдържа в три команди от менюто Insert:

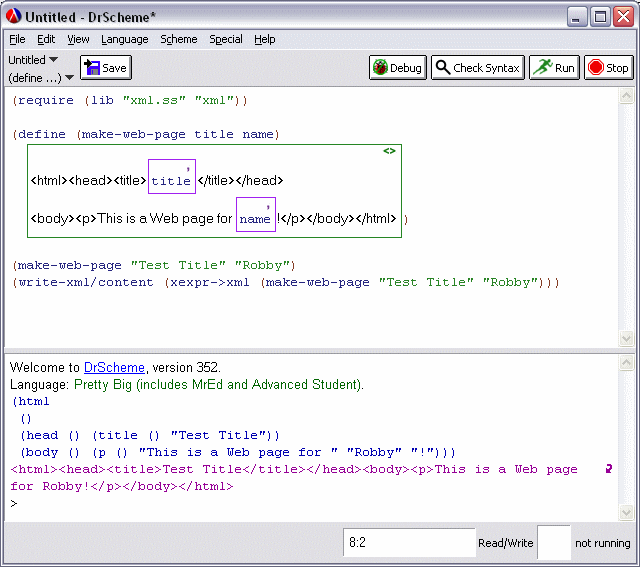
**Insert XML Box** - използвайте това, за да вмъкнете подредактор, в който може да въведете същинския XML - източник.

**Insert Scheme Box** - въведете Scheme - код в рамките на XML кутията, като използвате Scheme - кутия.

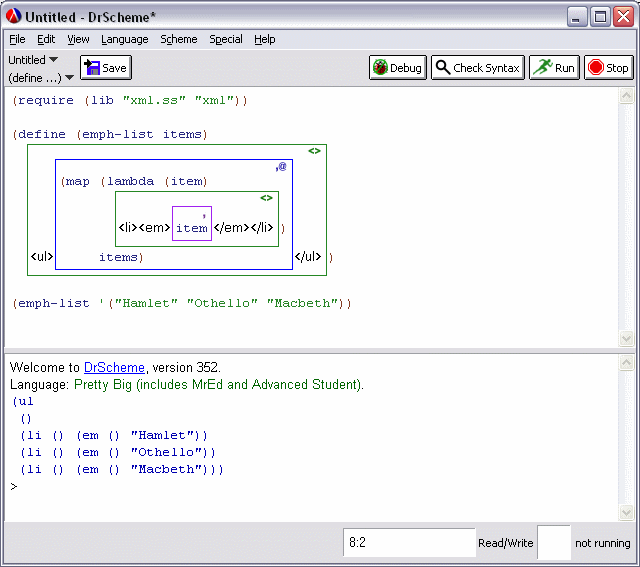
**Insert Scheme Splice Box** - тази кутия е подобна на Scheme - кутията, с изключение на това, че Scheme - изразите трябва да се изпълнят като списъци, които се свеждат до XML - съдържание.

Изходът е във формата на x - изразна форма.

Първия пример показва прост шаблон на Web - страница: функцията съдържа загравието и името на генерираната web - страница за определено лице. Първото извикване показва x - изразния изход, докато второто показва как потребителя може да конвертира това да същински XML синтаксис.



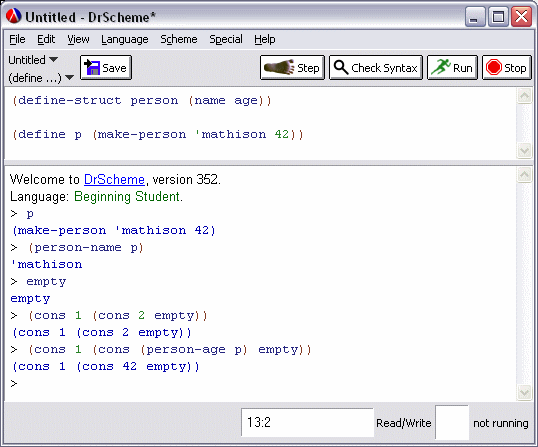
Втория пример показва употребата на Scheme Splice Box. Комбинацията от функции от по - висок ред, списъци и обръщане на резултатите е един изключително мощен инструмент за генериране на документи.



1. ***Разпечатване на стойности***

***Конструктор на обозначения***

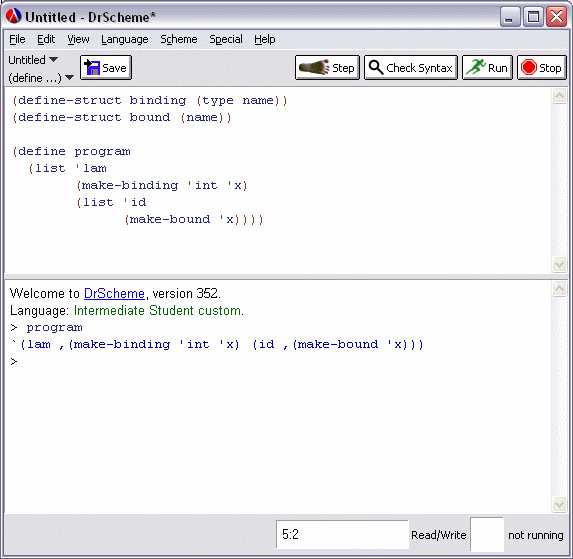
Интерактивния прозорец разпечатва стойности, използвайки конструктор на обозначения за съставни стойности, така че студентите да се научат да разбират това като стойности, а не като специален синтаксис. Опитът показва, че това помага на студентите да разберат тънкостите на обработката на списъци по - добре, отколкото когато се използват цитирания и **'** за стойностите на списъци.



***Квази - куод означения***

Има моменти, когато конструкторът за обозначения може да стане по - скоро тромав, например когато студентите използват списъци за представяне на програмите в курса на обучение. Поради тази причина Scheme предлага квази - куод означения за разпечатване на стойности.

Стилът на разпечатване може да бъде избран независимо от нивото на езика в повечето случаи (т.е. когато нивото на езика осиурява необходимите функции).



1. ***Преносимост***

Scheme е преносим по две причини:

* Той работи на всички платформи, включително Windows 95/98/NT/2000/XP, Macintosh и няколко дистрибуции на Unix и X- Windows. Scheme предлага по същество идентичен интерфейс на всички платформи, с незначителни промени, за да се изпълнят изискванията на всяка платформа.
* Програми, написани на Scheme се държат по един и същ начин на тези платформи. Това включва и програми, които използват графичните библиотеки. Напредналите програмисти могат да се възползват от конфигурацията на конкретната платформа, за да създават например приложения на Macintosh, които да се усещат като такива. Същото е и за приложенията на Windows.

1. ***Полезни пакети***

Scheme потребителите са допринесли с много полезни пакети, които може да откриете в хранилището:

http://planet.plt-scheme.org/

В допълнение, индивидуалните разработчици и авторите на учебници също разпространяват пакети. Повечето от тези пакети са във формат .**plt**. Това е многоплатформен самостоятелноинталируем формат, специфичен за Scheme. За да инсталирате .**plt** файл, независимо дали в интернет или в локална файлова система, използвайте **File** -> **Instal .plt File.**

1. ***Заключение***

Scheme е богата среда, с поддръжка на графични нитерфейси (какъвто е и самия Scheme), уеб - програмиране, XML, ActiveX (документацията може да не се инсталира на вашата платформа) и мн. др. На сайта

http://www.plt-scheme.org/

може да намерите още неща за Scheme.