Руководство по выполнению расчетной работы по курсу «Проектирование программ в интеллектуальных системах»

В следующем разделе я описываю выполнение 2-го этапа расчетной работы на примере задачи поиска одного из минимальных путей в неориентированном графе.

# Этап: Разработка программы решения теоретико-графовой задачи на языке программирования, предназначенном для обработки семантических сетей

## Задание

На этом этапе выполнения расчетной работы вам необходимо будет разработать программу на языке программирования, предназначенном для обработки семантических сетей, которая бы решала вашу теоретико-графовую задачу на основе формализации предметной области, проведенной на предыдущих этапах. На втором этапе вы должны были написать программу на С++ с использованием программной модели sc-памяти. Если на втором этапе вы написали программу с обработкой информации только в sc-памяти, то данный этап для вас будет очень простым.

В качестве тестов для написанной программы необходимо использовать тестовые примеры, которые вы сделали в ходе предыдущего этапа расчетной работы.

## Установка и настройка рабочей среды

Сперва мы установим и настроим рабочую среду для программирования на языке SCP (Semantic Code Programming) и запустим программу-пример поиска одного из минимальных путей. Всё описанное в этой главе программное обеспечение находится на кафедральном сервере info в папке \\Info\StudInfo\~Методическое обеспечение кафедры\~Учебные курсы\2 курс\ППвИС\1sem\Расчётная работа. Поэтому в дальнейшем я не буду указывать полный путь для программного обеспечения и исходных текстов, а буду ссылаться на эту папку. Нам пригодится модуль sc-core, который мы устанавливали на предыдущем этапе расчетной работы. Напомню, что я ставил его в папку c:\sc-core и именно этот путь буду использовать в дальнейшем. Но базу знаний с программой-примером необходимо обновить, поэтому удаляем папку c:\sc-core\examples\fs\_repo\_src и копируем папку fs\_repo\_src с сервера info в c:\sc-core\examples.

Среда разработки для языка SCP написана как плагин к платформе Eclipse, поэтому сначала нам необходимо поставить Java Development Kit. Берем с сервера info установщик jdk-6u29-windows-i586.exe и инсталлируем это ПО. Теперь берем архив eclipse-scpdev-3.6.2-win32.zip с сервера info и разархивируем его, например, в d:\tools\. После этого запускаем исполняемый файл d:\tools\eclipse\eclipse.exe.

С самого начала работы со средой Eclipse она попросит вас указать путь к рабочему пространству (workspace), в котором будет происходить создание проектов. Я указал путь d:\workspace (см. рисунок 2.1). Нажимаем OK.

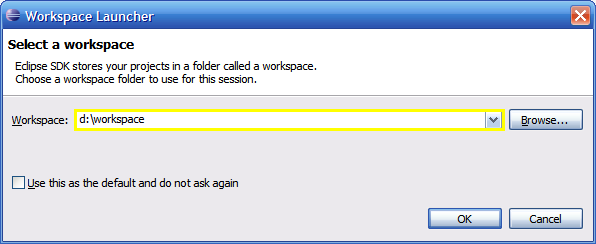


Рисунок 2.1 Выбор рабочего пространства

Среда Eclipse запустилась, и сразу же закройте вкладку Welcome. Теперь настроим среду выполнения для работы с проектами баз знаний. Для этого выбираем пункт меню Window -> Preferences. Выбираем пункты SCP -> SCP Enviroment. Там снимаем флажок и указываем путь к корневой директории модуля sc-core (в моем случае это c:\sc-core) как показано на рисунке 2.2. Нажимаем кнопки Apply и затем OK.

Пришло время создать проект с примером программы поиска минимального пути. Выбираем пункт меню File -> New -> Project и затем выбираем тип проекта SC Repository в папке SCP Development (см. рисунок 2.3). Нажимаем на Next. На этой странице мастера по созданию проекта (см. рисунок 2.4) вводим имя проекта wave\_find\_path и путь к папке с исходными текстами базы знаний c:\sc-core\examples\fs\_repo\_src (вспомните, что именно сюда мы копировали наш пример). Нажимаем кнопку Finish.

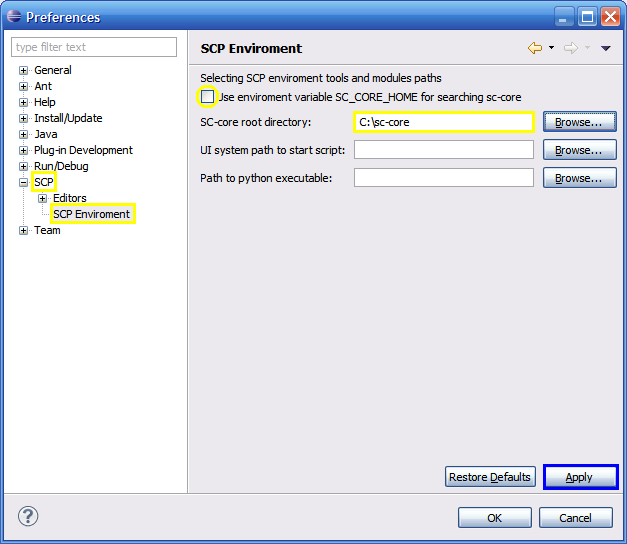


Рисунок 2.2 Настройка среды исполнения

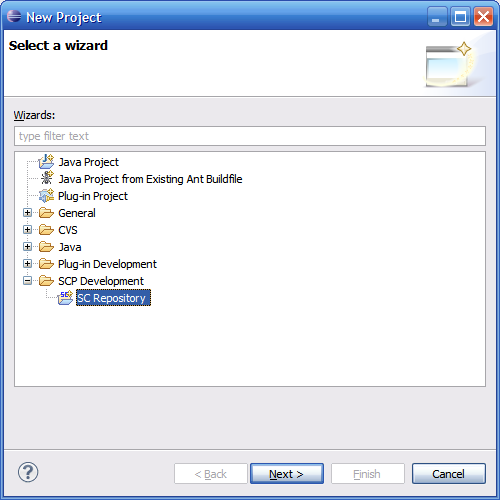


Рисунок 2.3 Выбор типа проекта

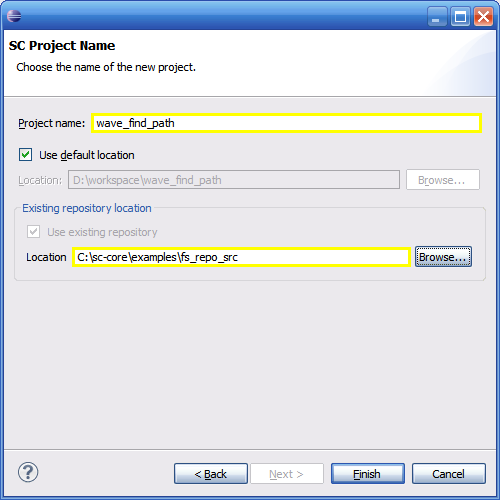


Рисунок 2.4 Настройка проекта при создании

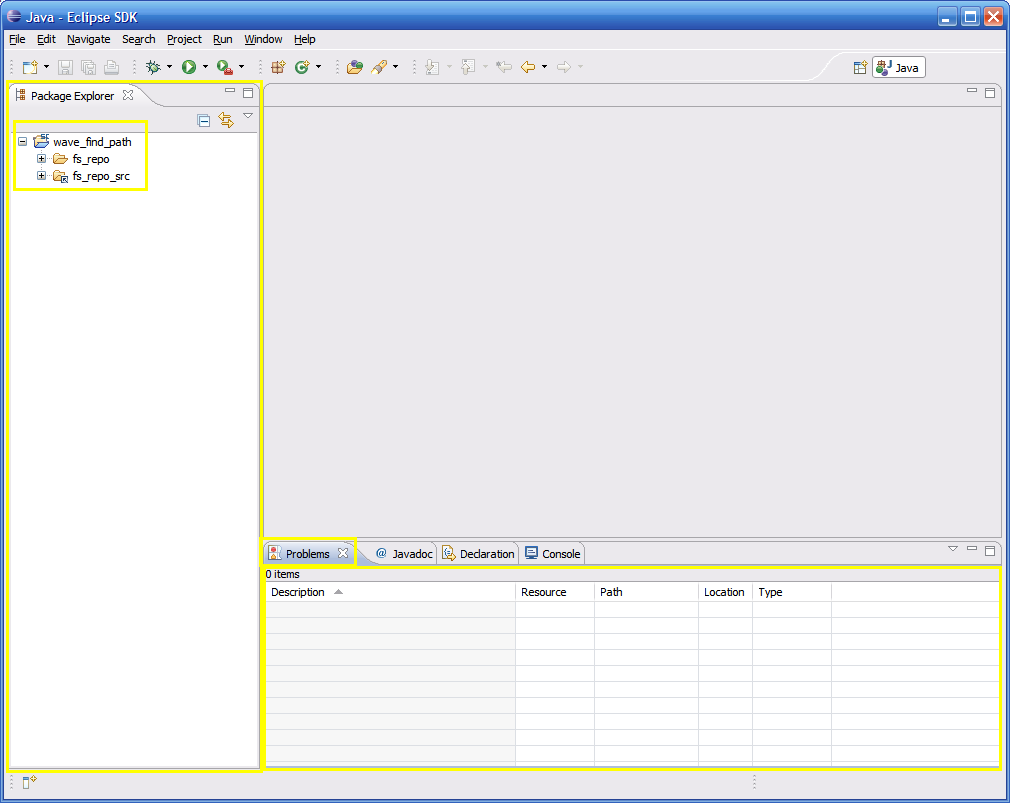


Рисунок 2.5 Проект wave\_find\_path создан и собран без ошибок

Проект создан и должен был быть автоматически собран (см. рисунок 2.5). Обратите внимание, что на вкладке Problems таблица пустая. Т.е. сборка произошла без ошибок. Еще необходимо обратить внимание на две папки в проекте wave\_find\_path. Они всегда присутствуют в проекте типа SC Repository. В папке fs\_repo\_src хранятся исходные тексты базы знаний (в том числе и программы), а папку fs\_repo является скомпилированной версией fs\_repo\_src. Содержимое этих двух папок мы рассмотрим в следующем разделе, а сейчас запустим программу-пример.

Для запуска проекта в Eclipse необходимо создать конфигурацию запуска. Вызовем диалог создания конфигурации запуска так, как показано на рисунке 2.6, или выбрав пункт меню Run -> Run Configurations….

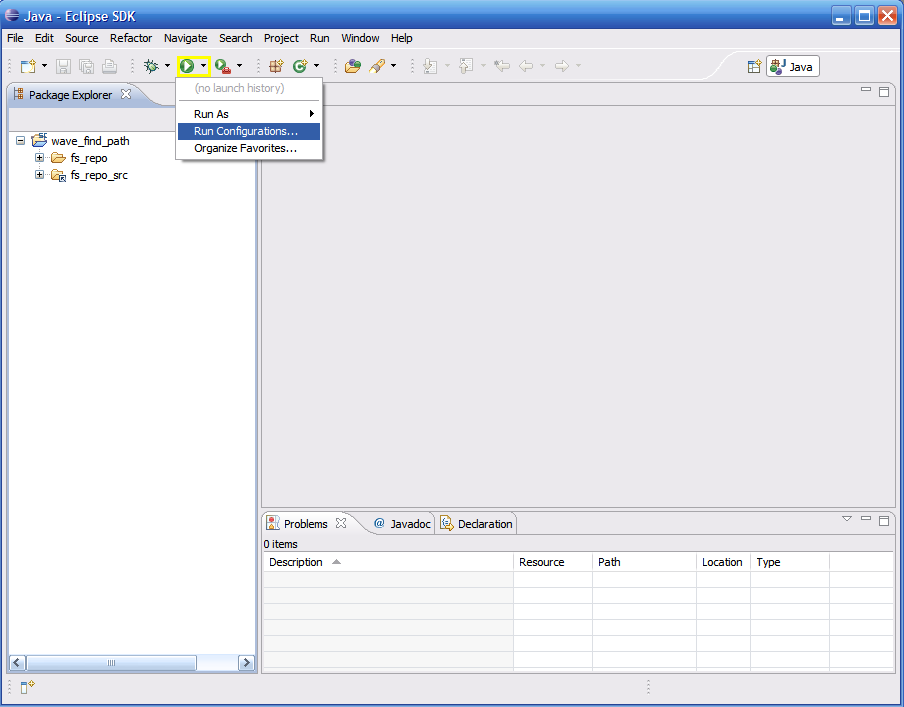


Рисунок 2.6 Вызов диалога создания конфигурации запуска

Тип конфигурации для запуска scp-программ из консоли называется «Run with start-pm». Нажимаем правой клавишей мыши на этом пункте и выбираем New (см. рисунок 2.7). Теперь необходимо настроить созданную конфигурацию.

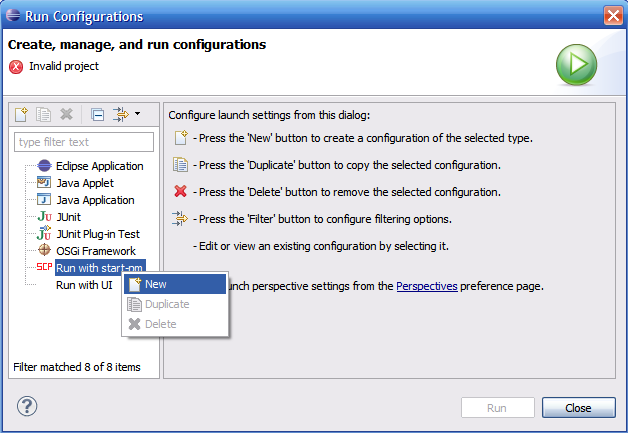


Рисунок 2.7 Создание конфигурации запуска из консоли

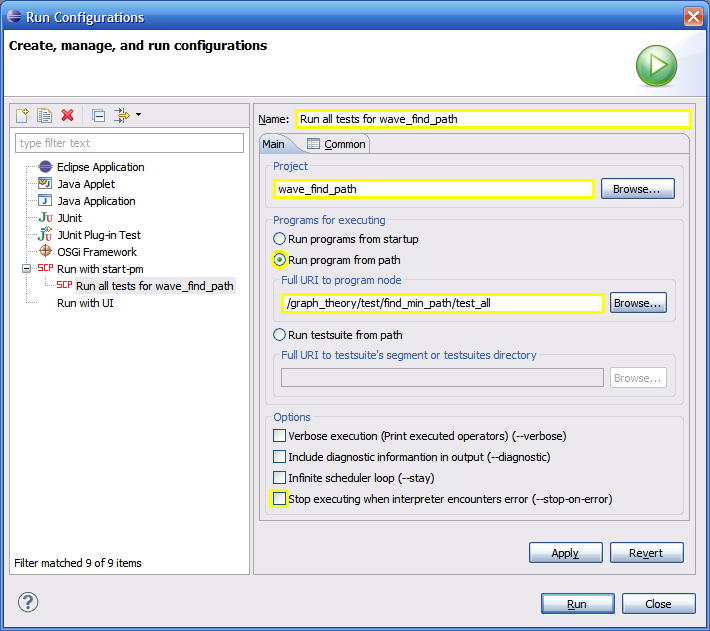


Рисунок 2.8 Настройка созданной конфигурации запуска

На рисунке 2.8 показана настройка конфигурации запуска. В поле «Name» введено имя конфигурации «Run all tests for wave\_find\_path» (имя может быть произвольным). В поле «Project» необходимо записать или выбрать из списка при помощи кнопки «Browse…» имя проекта, для которого будет работать эта конфигурация. В нашем случае это wave\_find\_path. В блоке «Programs for executing» выбираем «Run program from path», т.е. будем запускать программу по URI (данный способ запуска мы будем использовать всегда). В поле «Full URI to program node» записываем URI /graph\_theory/test/find\_min\_path/test\_all (как получается такой URI, мы рассмотрим в следующих разделах). **Обязательно** снимаем галочку с пункта «Stop executing when interpreter encounters error (--stop-on-error)» в блоке «Options». Нажимаем Apply и затем Run. Вы должны увидеть вкладку Console в нижней части экрана, в которую будет выводиться информация о выполнении тестовых примеров.

Так как полноценного отладчика scp-программ нет, то для отладки необходимо использовать конфигурацию запуска, которая показана на рисунке 2.9. В ней необходимо установить флажок «Verbose Execution (Print Executed Operator) (--verbose)». Нажимаем кнопки Apply и затем Run. Теперь во вкладке Console должен печататься полный пооператорный лог выполнения всех вызванных scp-программ.

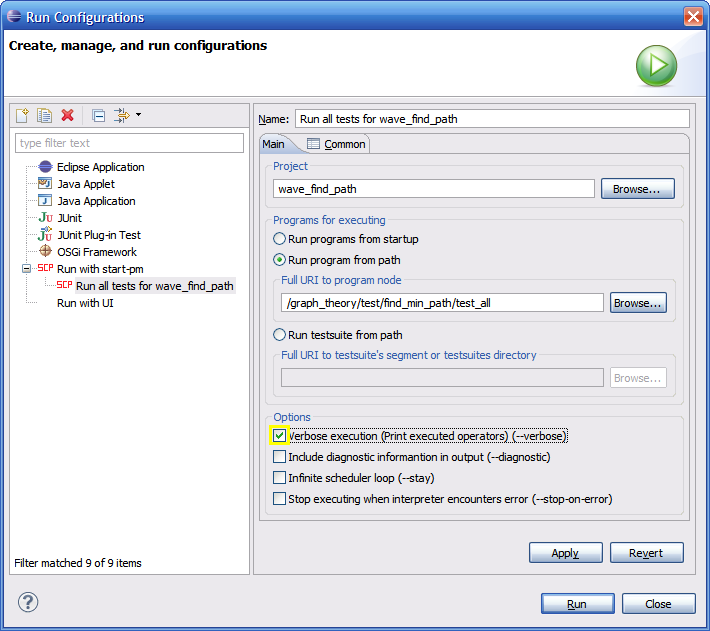


Рисунок 2.9 Конфигурация запуска из консоли с подробным выводом

Мы закончили настройку рабочей среды и теперь перейдем к описанию содержимого исходного и бинарного репозиториев базы знаний.

## Содержимое исходного и бинарного репозиториев базы знаний

Исходные тексты базы знаний (а к ним относятся и программы на SCP) хранятся в исходном репозитории, который обычно называется fs\_repo\_src. Обратите внимание, что в созданном нами проекте wave\_find\_path такая папка присутствует. Как вы уже знаете, на данном этапе существования технологии используется реализация sc-памяти с сегментной моделью. Есть sc-сегменты, в которых содержаться sc-элементы, и sc-директории, в которых содержатся sc-сегменты. Для удобства адресации этих трех видов объектов можно использовать URI (***Uniform Resource Identifier***). Если кто-то это подзабыл, то необходимо обратиться к документации по программной модели sc-памяти из руководства по выполнению 2-го этапа расчетной работы.

Исходный репозиторий компилируется в папку fs\_repo. Как вы заметили, структура sc-памяти с точки зрения сегментной модели имеет свойство древовидности. Это позволяет хранить содержимое sc-памяти на файловой системе один в один. URI / соответствует корень бинарного репозитория (папка с именем fs\_repo). Любой подпапке в бинарном репозитории fs\_repo соответствует sc-директория с тем же именем, а любому файлу в fs\_repo соответствует sc-сегмент с тем же именем. Причем URI sc-директорий и sc-сегментов будет записываться относительно папки fs\_repo, т.е. корня репозитория. Таким образом у нас есть исходный репозитория, из которого получается бинарный репозиторий, на основании которого формируется содержимое sc-памяти. Рассмотрим подробно исходный репозиторий.

Каждая папка исходного репозитория будет существовать и в бинарном репозитории, если в этой папке есть хотя бы один файл, который может быть оттранслирован в бинарную версию. Исходные файлы бывают трех типов:

* Тип файла, при котором для формирования сегмента используются имя файла и его положение в исходном репозитории. Например, fs\_repo\_src/graph\_theory/find\_min\_path.m4scp будет оттранслирован и будет соответствовать fs\_repo/graph\_theory/find\_min\_path в бинарном репозитории. Как видите, убирается только расширение. Такие исходные тексты могут быть следующего вида (определяется по расширению):
  + \*.scs – исходные текст на SCs (Semantic Code string). Линейная форма представления sc-тектов;
  + \*.m4scp – исходные тексты SCP-программ, транслируемые в SCs;
* Тип файла, для которого не формируется сегмент. Такие исходные тексты могут быть следующего вида (определяется по расширению):
  + \*.scsy – заголовочные файлы синонимом, которые могут быть подключены в scs-файлы и m4scp-файлы. В исходном репозитории есть особая папка include, в которой находятся все scsy-файлы. В некотором роде scsy-файл является аналогом заголовочного файла в С/С++.
* Тип файла, который не подлежит обработке. Все файлы, кроме перечисленных выше.

В следующих разделах мы рассмотрим форматы scs-, scsy- и m4scp-файлы.

## Язык SCs для представления sc-конструкций в линейном виде

В файлах с расширением scs должен содержаться текст на языке SCs. Так же, как существует графический язык для записи sc-конструкций SCg, существует и текстовый язык для записи sc-конструкций SCs.

Текст на языке SCs состоит из sc.s-предложений. Каждое sc.s-предложение должно заканчиваться точкой с запятой «;» и простейшим видом такого предложения является:

/\* Это комментарий \*/

element; // Это тоже комментарий

Такое scs-предложение будет соответствовать рисунку 2.10. Как видно из рисунка, приведенный sc.s-текст в sc-памяти будет выглядеть как константный sc-элемент неопределенного типа с идентификатором element.

D:\work\@2c1s\ППвИС_РР\Этап3\3.4 Язык SCs для представления sc-конструкций в линейном виде\1. Простое scs-предложение.png

Рисунок 2.10 SCg-аналог для простейшего sc.s-предложения

Предложения в SCs могут быть более сложными. Например:

node -> element;

Это соответствует SCg-тексту на рисунке 2.11.

D:\work\@2c1s\ППвИС_РР\Этап3\3.4 Язык SCs для представления sc-конструкций в линейном виде\2. Простое scs-предложение.png

Рисунок 2.11 SCg-аналог для более сложного sc.s-предложения

Как можно заметить, «->» используется для обозначения константной позитивной sc-дуги. Как видно из приведенного выше рисунка, sc-элемент node является константным sc-узлом, а не константным sc-элементом неопределенного вида. Это происходит потому, что из node есть выходящая sc-дуга, поэтому он является sc-узлом.

Аналогом рассматриваемого текста является:

element <- node;

Все то же самое, просто поменяли местами элементы и изменили направление стрелки. В таблице 3.1 представлен полный перечень форм записи различных видов sc-дуг на SCs.

Таблица 3.1 Соответствие SCg-изображения sc-дуги записи на SCs

|  |  |
| --- | --- |
| Изображение на SCg | Запись на SCs (направления слева направо и справа налево) |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_const_perm_pos.png | ->  <- |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_var_perm_pos.png | ->>  <<- |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_meta_perm_pos.png | ->>>  <<<- |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_const_perm_neg.png | />  </ |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_var_perm_neg.png | />>  <</ |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_const_perm_neg.png | />>>  <<</ |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_const_perm_fuz.png | ~>  <~ |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_var_perm_fuz.png | ~>>  <<~ |
| D:\work\@2c1s\ППвИС_РР\Этап3\scg\pair_meta_perm_fuz.png | ~>>>  <<<~ |

Предложения на SCs могут иметь еще более сложную структуру, чем было показано выше. Например:

node1, node2 -> element1, attr1\_: attr2\_: element2;

Этот sc.s-текст соответствует sc.g-тексту, показанному на рисунке 2.12. Как видите, sc-дуги проводятся от node1 и node2 и к element1, и к element2. А к sc-дугам, которые входят в element2, добавляются атрибуты attr1\_ и attr2\_ (для этого используется символ «:»).

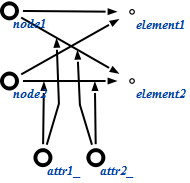


Рисунок 2.12 SCg-аналог для сложного sc.s-предложения c атрибутами

Для того чтобы задавать более удобно множества, можно использовать следующую запись:

{a, attr1\_: b, attr1\_: attr3\_: c};

Этот sc.s-текст соответствует sc.g-тексту, показанному на рисунке 2.13. Как видно из рисунка.

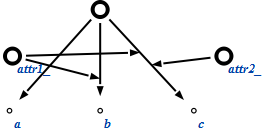


Рисунок 2.13 SCg-аналог для задания множества без идентификатора

## Язык программирования SCP и формат M4SCP для записи scp-программ

Первое знакомство с языком SCP и форматом записи scp-программ M4SCP начнем с написания программы «Hello world». Для этого сначала необходимо создать m4scp-файл. Выбираем пункт меню File -> New -> Other… и в открывшемся мастере выбираем «M4SCP Source File» (см. 2.14), а затем нажимаем на кнопку Next.

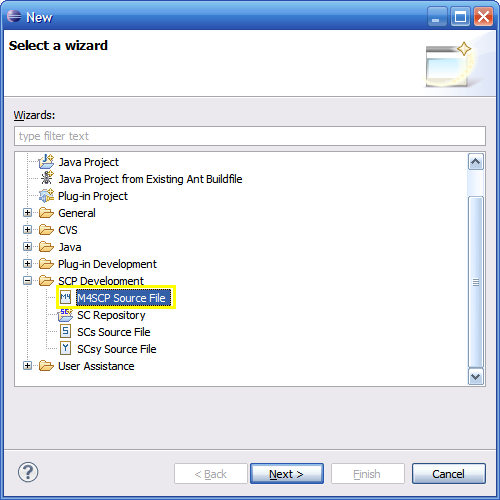


Рисунок 2.14 Запуск мастера создания M4SCP-файла

Мы будем создавать файл hello\_world.m4scp в папке fs\_repo\_src\lib, поэтому страницу мастера необходимо заполнить так, как показано на рисунке 2.15. Необходимо сказать, что в поле Templates можно выбрать шаблон содержимого создаваемого m4scp-файла. На рисунке 2.15 указано создание «M4SCP Program», т.е. scp-программы на M4SCP, которая не принимает параметром. Если выбрать шаблон «M4SCP Procedure», то будет создана scp-программа на M4SCP, которая принимает параметры. Если выбрать шаблон «Empty», то будет создан пустой m4scp-файл.

Теперь пришло время нажать кнопку Finish.

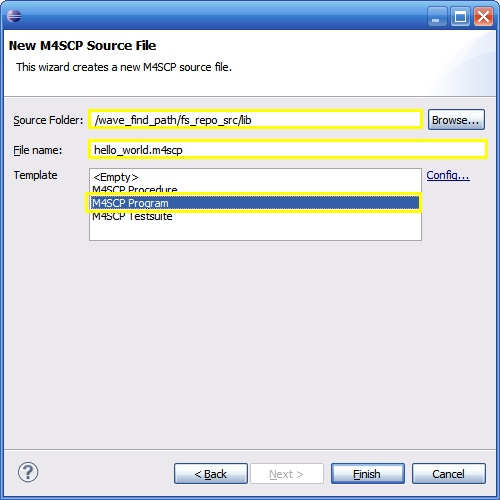


Рисунок 2.15 Мастер создания M4SCP-файла

Должен быть создан и открыт файл hello\_world.m4scp. Замените текущее имя программы program\_name на hello\_world и сохраните файл. Обратите внимание, что когда вы сохранили файл, компиляция проекта произошла автоматически.

На данный момент наша программа hello\_world состоит только из оператора return, т.е. возврата. Давайте попробуем ее запустить. Так как программа находиться в файле fs\_repo\_src\lib\hello\_world.m4scp и называется hello\_world, то полный URI к узлу программы в sc-памяти будет /lib/hello\_world/hello\_world. Осуществите запуск программы по URI /lib/hello\_world/hello\_world так, как было описано в разделе 2.2 (лучше не используйте старую конфигурацию запуска, а создайте новую).

Теперь давайте превратим наше программу в настоящую программу «Hello world». Для этого замените содержимое файла hello\_world.m4scp на приведенное ниже:

// Заголовочный файл с описанием синонимом для

// ключевых узлов языка SCP. Файл scp\_keynodes.scsy

// должен быть подключен в любом m4scp-файле.

#include "scp\_keynodes.scsy"

// Объявление программы hello\_world

program(hello\_world,

[[

// Секция констант scp-программы (здесь возможен любой sc.s-текст)

message =c= /"Hello world!!!"/;

]],

[{

// Секция переменных scp-программы

}])

// Здесь начинается тело scp-программы

// Оператор вывода содержимого sc-узла message на консоль

printNl([1\_: fixed\_: message])

// Оператор возврата из программы

return()

end // Здесь заканчивается тело scp-программы

Сохраните файл, и сборка будет проведена автоматически. Попробуйте опять запустить программу /lib/hello\_world/hello\_world. Вы должны увидеть в окне Console выведенную строку «Hello world!!!». Вот вы и написали вашу первую scp-программу.

Это руководство не является справкой по языку SCP и формату записи scp-программ M4SCP, поэтому вам необходимо взять из папки расчетной работы на сервере info справочный файл «scp\_guide.chm». В нем для вас представляют интерес следующие главы:

* Введение
* Базовые понятия языка
* Как пользоваться руководством
* Первая программа
* Структура scp-программы и scp-процесса
* Операторы языка SCP
* Способы ветвления scp-программы
* Протокол работы scp-программы
* Тестирование и отладка scp-программы
* Советы по программированию
* Оптимизация исходного текста
* Заключение

В справке по языку SCP не описана среда разработки на базе Eclipse, а описаны консольные утилиты для разработки. Так как вам необходимо использовать графическую среду разработки, описанную в данном руководстве, то просто не обращайте внимания на описание работы с консольными утилитами при чтении справки.

Еще по языку SCP можно почитать [официальную описание](http://www.ostis.net/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_%D0%B8_%D0%B4%D0%B5%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D0%BA%D0%B0_SCP) на сайте проекта OSTIS, но она написана формальным языком, и может быть сложна для понимания неподготовленным читателем. Поэтому всю необходимую информацию вам должна предоставить справка из chm-файла.

А теперь мы продолжим экспериментировать с программой hello\_world. Посмотрим как будет вести себя среда разработки при обнаружении синтаксической ошибки в тексте программы. Удалите «:» после атрибута fixed\_ в операторе printNl. Сохраните файл, чтобы произошла перекомпиляция проекта. В результате среда разработки должна вывести ошибку так, как показано на рисунке 2.16. Обратите внимание, что сейчас редактор находится на вкладке «M4SCP source code». Если вы попробуете перейти к месту ошибки, щелкнув два раза по ней в окне «Problems», то вас перекинет на вкладку редактора «Generated SCs code», где покажет строку, в которой найдена ошибка (см. рисунок 2.17). Так как M4SCP транслируется при помощи специального процессора в SCs, а затем компилируется, то приведенный выше способ является единственным для обнаружения и исправления синтаксической ошибки в m4scp-файле. Вкладка редактора «Generated SCs code» всегда находится в режиме «Только для чтения», поэтому чтобы исправить ошибку, вернитесь на вкладку «M4SCP source code» и поставьте двоеточие.

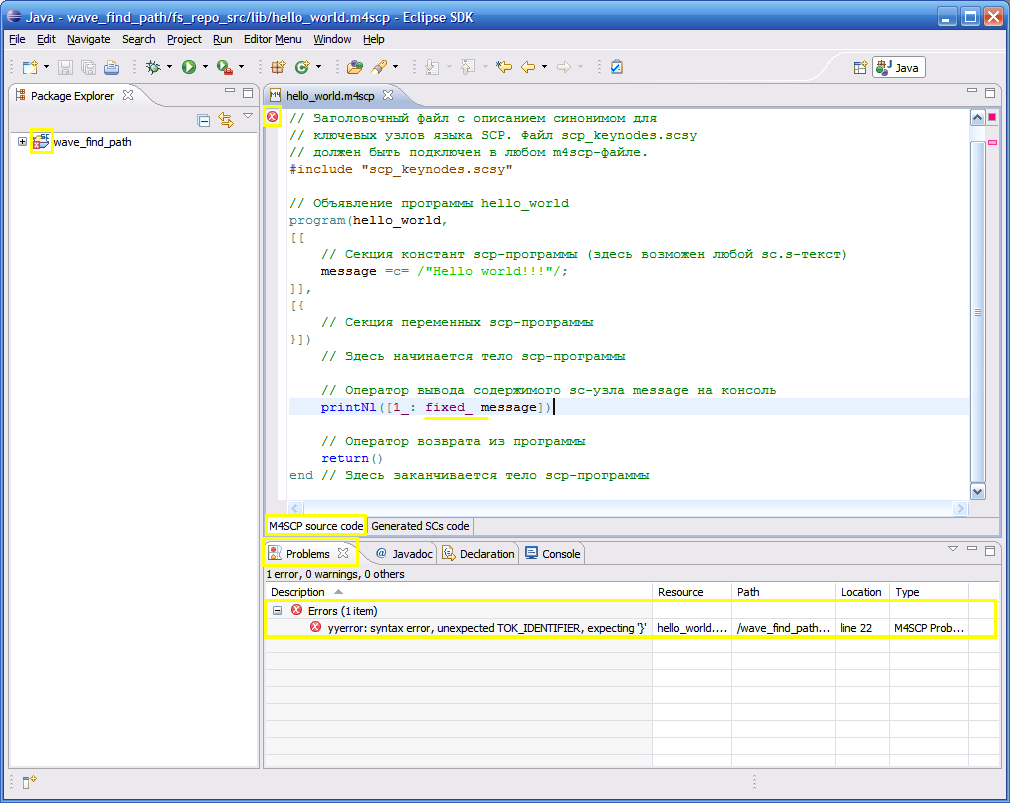


Рисунок 2.16 Синтаксическая ошибка в hello\_world.m4scp

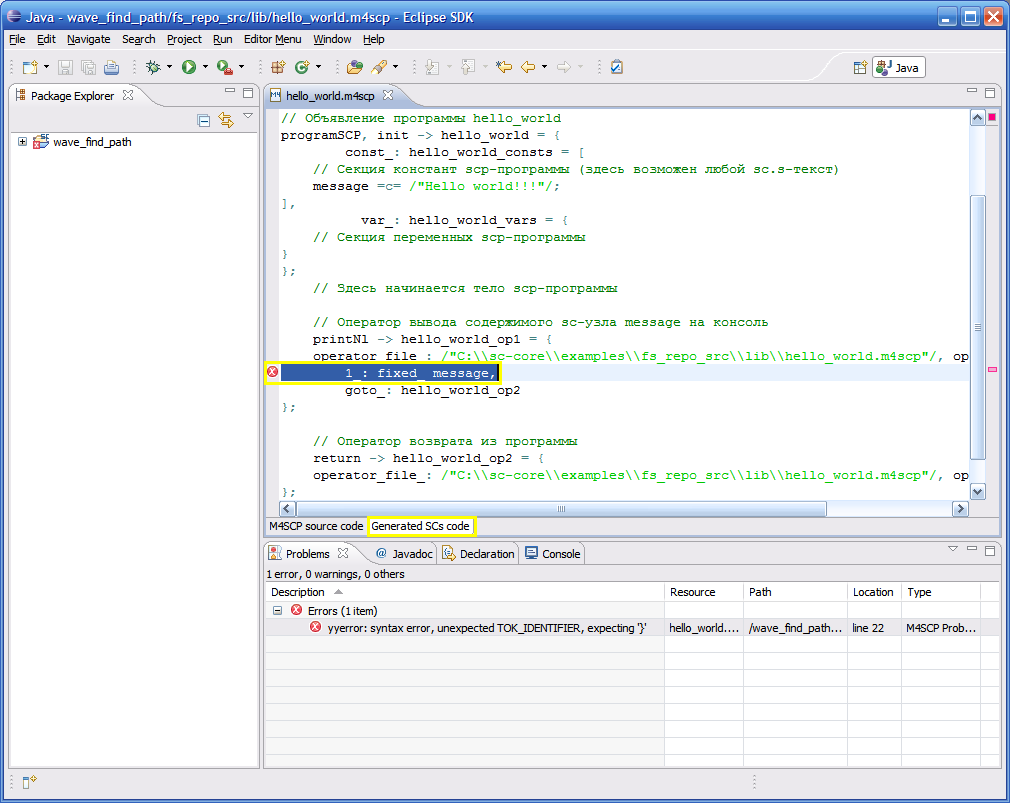


Рисунок 2.17 Синтаксическая ошибка в sc.s-трансляции hello\_world.m4scp

Давайте посмотрим, как будет вести себя среда при обнаружении ошибки на этапе выполнения scp-программы. Давайте добавим в секцию переменных переменную var1 и заменим в операторе printNl вывод константы message на вывод переменной var1. Сохраните и запустите программу hello\_world. Так как переменная var1 не имеет значения, то scp-интерпретатор должен вывести ошибку (см. рисунок 2.18). Чтобы перейти к оператору, при исполнении которого произошла ошибка, щелкните на выведенную ссылку в окне «Console».

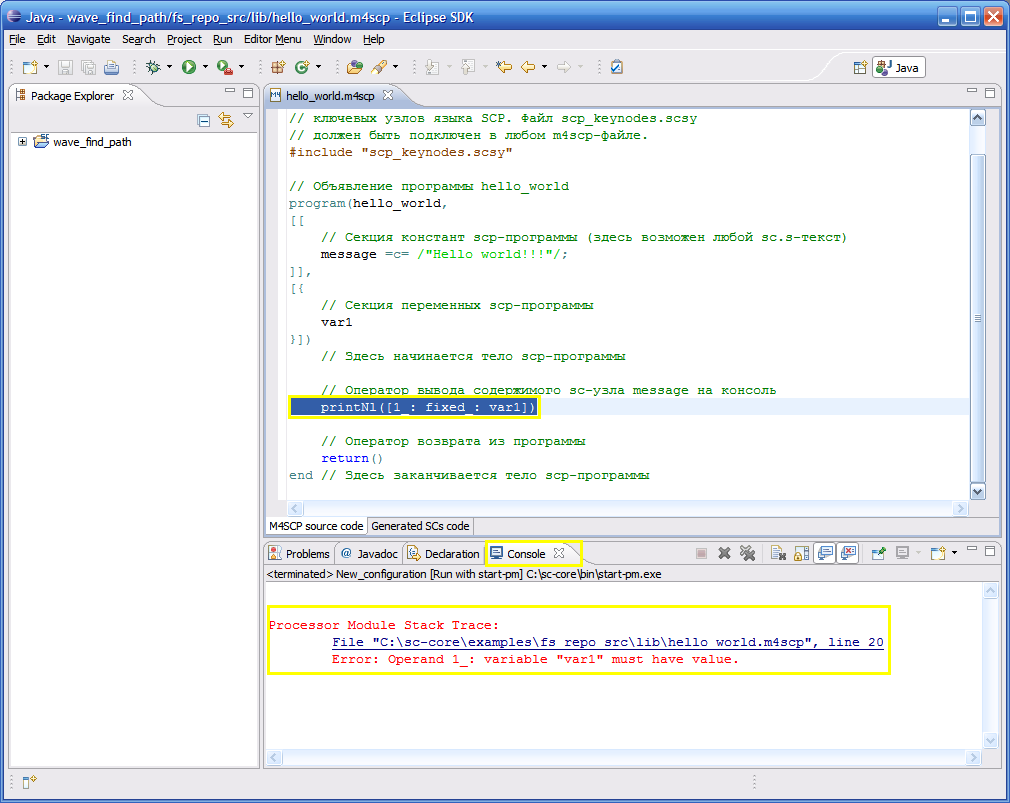


Рисунок 2.18 Ошибка на этапе исполнения

## Программа-пример поиска одного из минимальных путей в неориентированном графе

Исходные тексты всех scp-программ, которые относятся к реализации волнового алгоритма поиска одного из минимальных путей, находятся в папке fs\_repo\_src\graph\_theory. Каждая scp-программа находится в отдельном m4scp-файле с соответствующем именем. Все scp-программы подробно прокомментированы и полностью соответствуют аналогичным C++ функциям из 2-го этапа расчетной работы (вплоть до имен переменных и комментариев), поэтому не должно возникать проблем с тем, чтобы разобраться в том, что делают scp-программы и как они это делают. Файлом синонимом для всех scp-программ из директории graph\_theory является файл fs\_repo\_src\include\graph\_keynodes.scsy. Ключевые узлы базы знаний по теории графов объявлены в файле fs\_repo\_src\graph\_theory\keynode.scs. Обратите внимание, что в файле graph\_keynodes.scsy нет синонимом для этих ключевых узлов, поэтому при их использовании необходимо указывать полный URI ключевого узла (например, “/graph\_theory/keynode/undirected graph” для ключевого узла “undirected graph”).

Тестовые примеры для программы-примера находятся в папке fs\_repo\_src\graph\_theory\test. Неориентированный граф для каждого тестового примера описан в отдельно scs-файле (см. test1.scs, test2.scs, test3.scs, test4.scs, test5.scs), а сами scp-программы, которые вызывают find\_min\_path, записаны в fs\_repo\_src\graph\_theory\test\find\_min\_path.m4scp. В этом же файле есть scp-программа test\_all, которая запускает все тестовые примеры одновременно (вспомните, что в разделе 2.2 мы запускали именно ее по URI /graph\_theory/test/find\_min\_path/test\_all).

Программу-пример и тесты для нее можно и нужно использовать как шаблон-заготовку для выполнения этого этапа расчетной работы.

В директории fs\_repo\_src\lib находится небольшая библиотека scp-программ для обработки двусвязных списков (поддиректория list), обработки бинарных отношений (поддиректория relation), обработки множеств (поддиректория set), вывода на консоль (поддиректория print). Все эти scp-программы и их параметры прокомментированы. Для их использования необходимо подключить файл синонимов lib\_keynodes.scsy.

## Советы написанию scp-программ в формате M4SCP

Старайтесь писать каждую scp-программу в отдельном m4scp-файле.

Метка в m4scp-файле это просто идентификатор scp-оператора. Поэтому, если вы напишите две m4scp-программы или процедуры в одном файле, и в каждом из них будет метка с одним и тем же именем, то это может привести к необъяснимому поведению программ при работе. Тоже самое можно сказать и про константы. Константы с одинаковым именем в рамках одного m4scp-файла будут склеены.

При создании конфигурации запуска в Eclipse обязательно снимите флажок «Stop executing when interpreter encounters error (--stop-on-error)».

Необходимость этого совета вызвана ошибкой в текущей версии scp-интерпретатора. Если не последовать этому совету, то при возникновении ошибки на консоль будет выводиться невразумительное сообщение «Stopped PM due to ERROR\_STATE», а не приятный глазу стэк вызовов с понятным сообщением об ошибке.

Если вы объявили параметр в секции параметров, то необходимо объявить переменную с таким же именем в секции переменных.

Параметр всегда должен быть продублирован в секции переменных.

Старайтесь выделять в вашей m4scp-программе или процедуре максимально возможно число подпроцедур.

Этот совет вызван тем, что тестирование более мелких подпроцедур по отдельности позволяет более быстро выявить ошибку. Так же это способствует написанию менее запутанного кода, так как циклы и условные конструкции в M4SCP необходимо реализовывать через переходы по меткам.

Следите за абсолютными URI констант в параметрах scp-программ.

Например, вы решили вывести множество по полному идентификатору:

#include "scp\_keynodes.scsy"

#include "lib\_keynodes.scsy"

// …

callReturn([

1\_: fixed\_: print\_set,

2\_: {[ 1\_: “/test/Set1” ]}

])

Предположим, что URI “/test/Set1” указывает на несуществующий sc-элемент. При исполнении такого кода scp-интерпретатор из-за ошибки просто завершится и не выведет никакого сообщения об ошибке.

Всегда включайте используемые sc-элементы (константы, scp-программы) в секцию констант m4scp-программы или процедуры.

Процесс scp-интерпретатора открывает для всех констант соответствующие им сегменты. Поэтому, если вы используете sc-элемент, который не включен в секцию констант, программы может работать «странным» образом. Например, из sc-элемента есть исходящие sc-дуги, но, так как он не включен в секцию констант, сегмент, где находятся эти sc-дуги, закрыт. А это значит, что вы не сможете найти ни одной исходящей sc-дуги.

Если что-то не работает, попробуйте осуществить полную пересборку проекта.

Выберите пункт меню Project -> Clean… (см. рисунок 2.19), там выберите «Clean all projects» и нажмите OK.

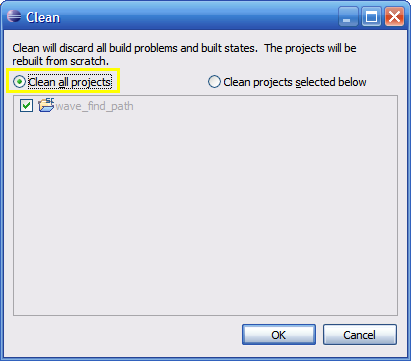


Рисунок 2. Полная пересборка проекта

Если что-то не работает, то вы всегда можете обратиться ко мне за советом.