электронное научно-техническое издание
НАУКА и ОБРАЗОВАНИЕ
Эл № ФС 77 - 30569. Государственная регистрация №0421100025. ISSN 1994-0408

Методические приемы контроля знаний по дисциплинам программирования

77-30569/281576

12, декабрь 2011

Большаков С. А.

УДК 004.423

МГТУ им. Н.Э. Баумана
chernen@bmstu.ru

Введение

Большое место в методической работе преподавателя вузов занимает подготовка вариантов заданий (лабораторных работ, домашних заданий, контрольных и курсовых работ). В современных условиях, при больших группах студентов и потоках, число таких вариантов велико. Кроме того, нужно быть готовым к тому, чтобы в каждом новом цикле дисциплины и для разных потоков студентов, иметь множества непересекающихся заданий и их вариантов. Это диктуется, к сожалению, несомненным “прогрессом” в области информационных технологий и средств коммуникации.

Важной задачей в этом направлении является повышение эффективности и качества контроля знаний, полученных студентами, полнота этой проверки и ее оперативность. В данной статье рассмотрены некоторые методические приемы контроля знаний, основанные на концепции представлений [1]. Эти приемы, с одной стороны, должны способствовать решению перечисленных проблем, а, с другой стороны, базируются на формализованном и методически обоснованном подходе. Далее в статье излагаются такие приемы применительно к дисциплинам подготовки по программированию.

Основные понятия

Процессы преподавания и проектирования программ тесно связаны с понятием представления введенным в [1]. В этом контексте, под представлением (обозначим его Р) понимается взаимосвязанная совокупность структурированной информации, которую можно рассматривать как единое целое. Информация должна быть представлена в понятном и обозримом виде. Такая информация (представление) рассматривается в определенном аспекте и может служить как этапом в области восприятия знаний, так и

результатом этапа проектирования программ. Представления могут быть классифицированы по разным признакам [1], основываться на соответствующем языке, быть четкими и размытыми. Применительно к процессу обучения, формирование четких и запоминающихся представлений в заданной области является главной задачей. При обучении программированию, важной задачей является не только восприятие и построение нужных представлений, но и получения навыков формирования новых, связанных с другими, в соответствии с выполняемым заданием.

Важнейшими видами представлений являются: графические (или визуальные – схемы и чертежи) представления, текстовые представления (документация), текстовые программные представления и электронные представления [1]. В нашем случае важно, что для описания представлений может быть использован формализованный язык, что обеспечивает, в конечном счете, четкость описания представлений и контроль выполнения задания преподавателем. Примерами таких представлений в области программирования и преподавания программирования являются: формализованные описания алгоритмов, блок схемы программ (графические), программы на псевдоязыках и языках программирования, диаграммы классов (программы), документация на программы и многое другое.

Другим важнейшим понятием, которое может быть положено в методику проверки знаний является переход между представлениями [1]. Переход от одного представления к другому, в общем виде, является, по сути, сложным преобразованием информации, он не всегда может быть формализован и основывается на навыках, знаниях и значительных интеллектуальных усилиях. Именно знания, навыки и возможность использования интеллектуального потенциала является целью обучения студентов предмету, а контроль результатов обучения сводиться к проверке выполнения заданий, включающих данные элементы. Другими словами, целенаправленный контроль результатов обучения студентов тесно связан с их умением проводить преобразования представлений (в нашей терминологии) и осмысленное их восприятие. В тех случаях, где возможно, целесообразно опираться на формализованные описания представлений и алгоритмы их преобразования.

Переходы также могут быть классифицированы по разным признакам [1]. Нас в данной статье интересуют только некоторые классов переходов: по направлению, по четкости и по числу шагов. По направлению выделяются: прямой и обратный переходы между представлениями. По четкости: четкие (формализованные) и нечеткие. По числу шагов выделяются одношаговые и многошаговые переходы. Свойство прямого и обратного перехода базируется на понятии предшествования представлений. Если первичное представление должно быть получено ранее при получении знаний или разработке программ, то такой переход следует считать прямым. Например, необходимо

сначала, разработав алгоритм, формализовать его в виде блок-схемы (или другого формального описания), а только затем приступать к разработке программы. Надо отметить, что можно говорить о преобладающем или логически обоснованном предшествовании, так как в жизни иногда бывает и наоборот.

Прямой переход между представлениями P1 и P2 схематично показан на рисунке 1.

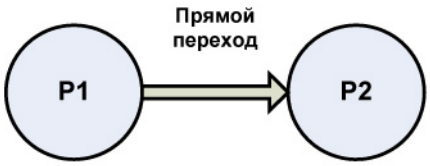

Рис. 1. Прямой переход между представлениями P1 и P2.

Понятие обратного перехода (рис. 2) между представлениями также является очень важным и, в первую очередь, для процессов контроля знаний, понимания алгоритмов построения представлений и использования их на практике. Например, если студент получит задание по разработанной и отлаженной программе построить блок-схему, то он должен проявить не только знания языка программирования и правил построения блок-схем, но и умение установить соответствие между элементами программы и графическими элементами блок-схемы для конкретного случая. Другими словами, он должен выполнить компиляцию, применяя необходимые знания, навыки и мыслительные действия.

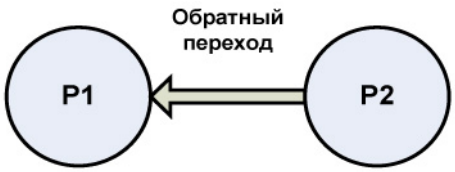

Рис. 1. Обратный переход между представлениями P1 и P2.

Четкие переходы межу представлениями (без потери информации) в общем случае возможны только между представлениями, описываемыми на формализованных языках [1]. Многошаговость представлений предполагает, как минимум, выполнение нескольких шагов преобразований. В этом случае не всегда может быть соблюдено условие четкости и формализации перехода. Особенности такого перехода для контроля знаний мы рассмотрим ниже. Ниже показан рисунок 3 для многошагового перехода. Такой переход можно наблюдать, например, при разработке комплекта документации на программы, включающий в себя разные взаимосвязанные документы: техническое задание, техническое описание, руководство пользователя, программа испытаний и т.д.

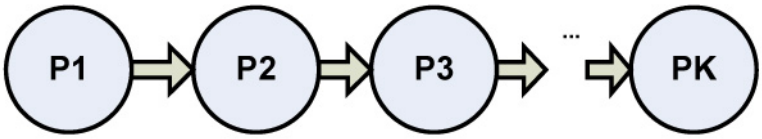

Рис. 3. Многошаговый переход.

Рассмотрим теперь, как представленные выше понятия и положения могут быть использованы для некоторых частных случаев подготовки заданий для студентов при преподавании программирования и контроле их знаний.

Методические приемы и задания, основанные на переходах

В первую очередь можно отметить, что задания могут быть построены на построении самих представлений. Это очевидно и используется чаще всего. Например, в задание входит разработка программы или диаграмм классов. Недостатком такого подхода является значительная длительность процесса решения задачи и разнообразность вариантов решения (разные переменные, разные алгоритмы и т.д.). Поэтому контроль знаний студентов становиться длительным и трудоемким процессом. Другой вариант заданий заключается в том, что студенту предоставляется в качестве задания готовое представление, описанное в обобщенном виде, и предлагается построить переход от этого представления к другому представлению. В этом случае первоначальное представление может быть написано (или изображено графически) на упрощенном языке (в нотации формальных схем), причем в нем должны быть отображены самые существенные элементы. Для подобных заданий используются как прямые, так и обратные переходы. На основе такого подхода можно выделить следующие типы заданий:

- Произвести изменения в P1, включая добавление, удаление и модификацию, на основе задания преподавателя, сформулированного устно и письменно;
- Произвести изменения в P2, включая добавление, удаление и модификацию, на основе задания преподавателя, сформулированного устно и письменно;
- Построение прямого перехода (задано только P1), нужно построить P2;
- Построение обратного перехода (задано только P2) , нужно построить P1;
- Внесение контрольных изменений в P1 и отображение их в P2 (заданы P1 и P2);
- Удаление элементов из P1 и отображение изменений в P2 (заданы P1 и P2);
- Добавление элементов в P1 и отображение изменений в P2 (заданы P1 и P2);
- Внесение контрольных изменений в P2 и отображение их в P1 (заданы P1 и P2);
- Удаление элементов из P2 и отображение изменений в P1 (заданы P1 и P2);
- Добавление элементов в P2 и отображение изменений в P1 (заданы P1 и P2);

Если учесть, что сами P1 и P2 могут быть разнообразными, то легко оценить число возможных вариантов, которые без труда можно сгенерировать для контрольных заданий. Хотя это не главное в поставленной задаче. Важно то, что можно выполнить проверку знаний, умений и навыков студента относительно быстро и при минимальных затратах времени и студента и преподавателя.

Понятия скелетных программ и описаний диаграмм

Выше было отмечено, что для четкого перехода между формируемыми представлениями необходимо иметь формальные языки описания программ или стандартные нотации оформления графических представлений (правила рисования диаграмм классов, правила формирования блок-схем и т.д.). Задачу можно еще более упростить, если для каждого отдельного случая использовать подмножества языков и правил, а точнее некоторые понятные псевдоязыки описаний. Во многих случаях это не трудно. Так для описания представлений программы, основанной на блок-схеме можно использовать так называемую скелетную структуру программы на псевдоязыке. В этом псевдоязыке отображаются основные действия, характерные для алгоритмов (процессы, циклы, условные конструкции, переключатели и т.д.), а остальные элементы (например, описания переменных и операторы) могут быть опущены. Аналогично делается и для диаграмм классов и их описаний на языках объектного программирования. Здесь выделяются сами классы, связи наследования и некоторые другие, а все остальное опускается. В результате в программе мы можем получить, так называемую скелетную структуру классов. Для заданий, связанных с другими проверками можно опустить и другие элементы, оставив те, знания о которых подлежат проверке. Здесь детально не описываются такие формальные языки описания. Их структура и применение понятно из приведенных ниже примеров. Далее приводятся некоторые примеры контрольных заданий студентам, основанные на методических приемах, изложенных выше.

Примеры заданий для контроля по программированию

На рисунке 4 представлены сразу два вида заданий. Они основаны соответственно на прямом и обратном переходах между представлениями блок-схемы (P1) и скелетной структуры программы (P2). Для упрощения понимания и сокращения места на рисунке оба перехода показаны вместе (см. стрелки).

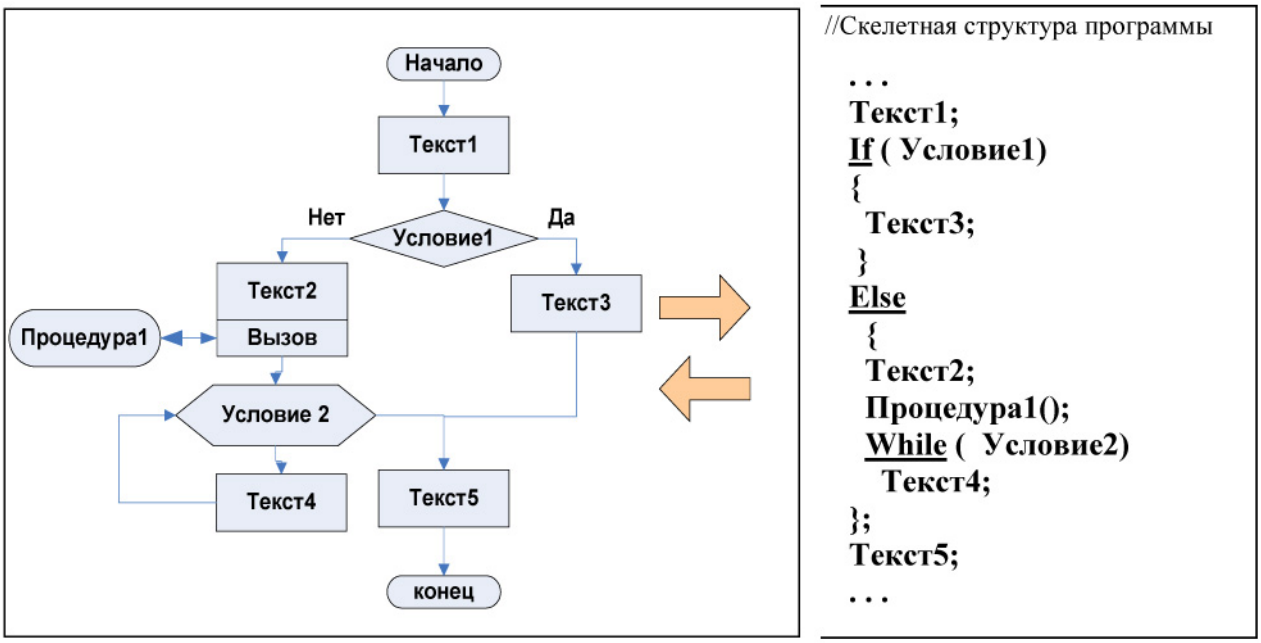

Рис. 4. Два вида заданий, основанных на прямом и обратном переходах.

Стрелки показывают направление преобразования (перехода), которое необходимо выполнить. В первом виде задании на основе обобщенной блок-схемы необходимо построить скелетную структуру программы на упрощенном псевдоязыке (If, Else, While и т.д.). Вместо конкретного кода программы записаны слова: “Текст1”, “Текст2” и т.д. Вместо условий: “Условие1”, “Условие2” и т.д. Во втором виде заданий студенту дается только скелетная структура программы, на основе которой он должен самостоятельно построить блок-схему ей соответствующую. В качестве дополнительных заданий, можно сформулировать вопросы такого вида: “Как измениться программа, если удалить фрагмент Текст3?”, или “Как измениться блок-схема, если добавить в цикл фрагмент Текст6?” и т.д. Многолетний опыт применения таких заданий показывает, что студенты, подготовившие тему, выполняют такое задание за 5-10 минут, включая даже и перерисовывание заданий в экзаменационные листы. Более слабые студенты, используя примеры таких заданий, легче воспринимают пройденный материал и в результате также справляются с подобными заданиями, хотя в начале пути подготовки они вообще не были готовы к таким задачам.

Примеры заданий для контроля знаний по проектированию систем классов

Ниже (рис. 5) приведен другой пример заданий, основанный на переходах между представлениями. Здесь слева представлена обобщенная диаграмма классов, в которой отображаются классы и связи: наследования, дружественность и включения (вложенность), а справа скелетное описание классов.

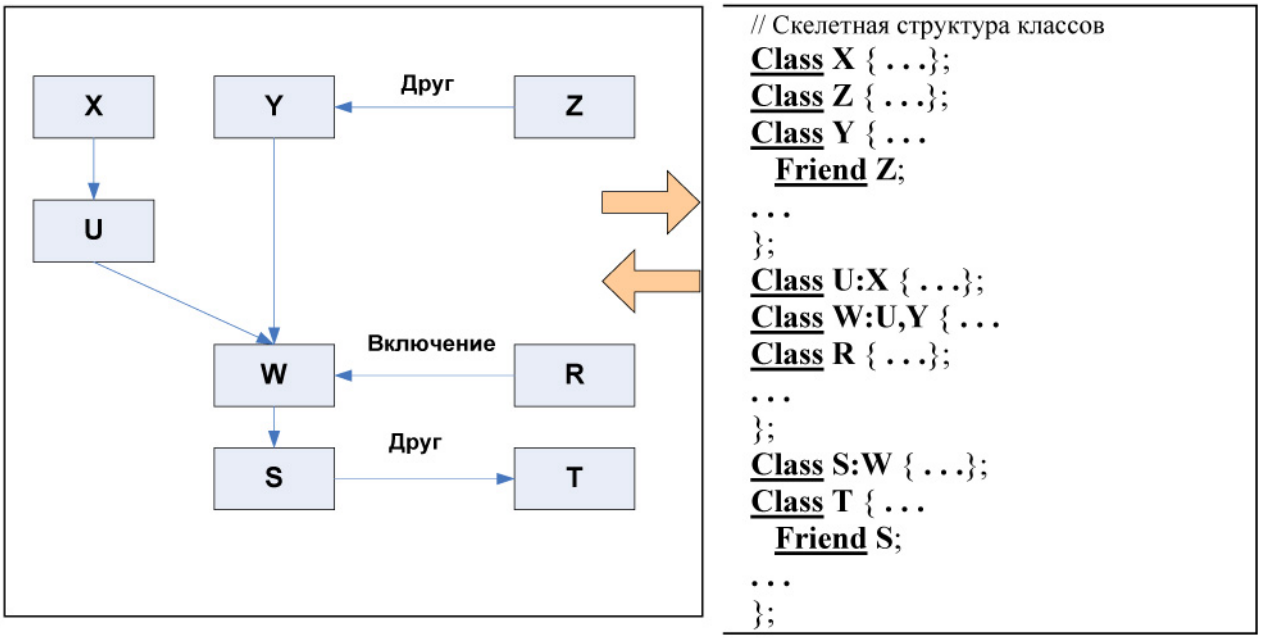

Рис. 5. Пример заданий, основанный на переходах между представлениями.

Задание заключается в том, чтобы на основе диаграммы построить скелетную программу описания заданных классов. Содержание классов здесь не отображается (заменяется многоточием). Дружественные связи помечаются на диаграмме текстом (“Друг”), а в программе friend, причем стрелка направлена от дружественного класса. Включение (вложение – описание класса в классе), также помечается текстом (“Включение”). Стрелка указывает на включающий класс. Связи наследования отображаются типичным способом, характерным для многих языков с классами (для заданий предпочтительнее использовать стрелки направленные от базового класса к наследнику). Студент должен не только правильно построить скелетную структуру описания классов, но и правильно задать последовательность этих описаний. В случае обратной задачи, студенты на основе скелетной структуры классов должны построить диаграмму классов, используя набор простых правил, перечисленных выше. Здесь, аналогично, можно сформулировать множества вопросов, связанных с изменением, добавлением фрагментов диаграммы или скелетной структуры классов.

На основе данных методических приемов, можно без особого труда разработать задания для контроля знаний студентов и различных представлений программ. Это может касаться: структур данных, структур объектов. Построения блок-схем, на основе текстового описания алгоритмов и других формализованных представлений. Кроме того, важным является то, что при таком подходе число вариантов заданий велико, они легко могут быть сформулированы. Практика показывает, что использование данных приемов способствует качественному контролю знаний студентов и лучшему усвоению пройденного материала курсов.

Методические приемы задания для множества представлений

Выше было отмечено, что для контроля знаний может быть использован многошаговый переход. На следующем примере (см. рис. 6) показано, в каких случаях это возможно. Студент, выполняя курсовую работу, по программированию, разрабатывает, в соответствии с требованиями, комплект документации на программу. Он включает следующие документы: технические задание (ТЗ), техническое описание (ТО), руководство пользователя (РП), программу методику испытаний (ПМИ), листы курсовой работы, листинг программ (ЛП) и т.д. Все документы, а это множество представлений {P}, взаимосвязаны. Во время защиты курсовой работы комиссия может, проверяя знания студента, предложить ему отметить (например, галочкой) во всех документах какую-нибудь деталь (например, пункт ТЗ), затрагивающую весь проект. В данном случае выполняется комплексная горизонтальная проверка знаний и проверка самостоятельности выполнения задания студентом. Студенту, аналогично, можно дать задание внести изменения в проект, добавить или удалить отдельный элемент любого документа (P_i). Проверяющим нужно провести сквозной контроль внесенных изменений по всем документам (по множеству - {P}). Практика использования такого приема, показывает, что в итоге студенты лучше делают проекты и готовятся к их защите, зная какие методы проверки их ожидают.

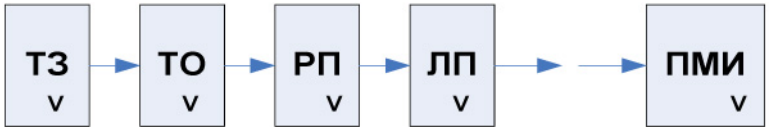

Рис. 6. Пример задания с многошаговым переходом.

Заключение

В данной статье рассмотрены методические приемы контроля знаний при обучении программированию в вузе. Использование их имеет ряд преимуществ: можно сформулировать большое число заданий одинаковой сложности; время выполнения таких заданий невелико; выполняется проверка не только фактических знаний, но и проверка умений формирования новой информации; время проверки результатов решений снижается, за счет формальности описаний; появляется возможность автоматизированного выполнения и проверки заданий.

Литература

1. Статья. Большаков С.А. Проблемы построения эффективных и безопасных прикладных программных систем. МГТУ им. Баумана. Каф. Сб. научных трудов. Вып.1 2000г.

2. Методические материалы на сайте: iu5.bmstu.ru. Путь к сайту “Преподаватели”-> “Большаков С.А.”

electronic scientific and technical periodical
SCIENCE and EDUCATION
EL № FS 77 - 30569. №0421100025. ISSN 1994-0408

Methodological ways of knowledge control in programming disciplines

77-30569/281576

 # 12, December 2011

Bol'shakov S.A.

Bauman Moscow State Technical University
chernen@bmstu.ru

The article covers students' knowledge control methods on a formalized approach basis. On the basis of ideas concepts as scope of information, and transition between them variants of students' knowledge control methods are created in connection with disciplines which teach programming. Examples of tasks are given, and advantages of this approach are marked out. Special features of using various control methods are considered.

Publications with keywords: presentation, students, knowledge monitoring, pass, variation of tests, examples of programming tasks, conduction of knowledge monitoring
Publications with words: presentation, students, knowledge monitoring, pass, variation of tests, examples of programming tasks, conduction of knowledge monitoring

Reference

1. Bol'shakov S.A., in: Collection of scientific works, MGTU im. Baumana – *BMSTU, Iss. 1, 2000.*