Игорь Геннадьевич Головин
Андрей Викторович Столяров

Мультипарадигмальный подход к преподаванию программирования и роль свободного ПО

Настоящий документ представляет собой тезисы доклада, сделанного на II конференции разработчиков свободных программ «На Протве» 27 июля 2005 года. Просьба при цитировании ссылаться на версию, опубликованную в сборнике тезисов докладов конференции с использованием следующей библиографической информации:

> Игорь Головин, Андрей Столяров. **Мультипарадигмальный подход к преподаванию программирования и роль свободного ПО.** // Тезисы докладов II конференции разработчиков свободных программ «На Протве», г. Обнинск, июль 2005 г.

Для пользователей LaTeX приводится библиографическая информация в формате `bibtex`:

```
@INCOLLECTION{golovin:teaching2005,
        AUTHOR={{Игорь Головин and Андрей Столяров}},
        TITLE="Мультипарадигмальный подход
               к преподаванию программирования и
               роль свободного ПО.",
        BOOKTITLE = "Тезисы докладов II конференции
               разработчиков свободных программ {{"<На Протве">}}",
        MONTH = "июль",
        YEAR = "2005",
        PAGES = "114--120",
        ADDRESS = "Обнинск"
}
```

Мультипарадигмальный подход к преподаванию программирования и роль свободного ПО

Игорь Головин, Андрей Столяров
МГУ им. Ломоносова, ф-т ВМиК

Аннотация

Рассматриваемый подход заключается в одновременном преподавании различных парадигм программирования (логического, функционального программирования и т.п.) на возможно более ранних этапах обучения. Такой подход более эффективно и гибко развивает у обучаемых алгоритмическое мышление по сравнению с традиционным подходом, основанным исключительно на императивном программировании.

Преподавание альтернативных парадигм программирования выдвигает ряд требований к используемому в процессе обучения программному обеспечению, прежде всего – к системам программирования и операционным системам.

Утверждается, что этим требованиям наиболее полно отвечают свободно распространяемые системы программирования и операционные системы семейства Unix, среди которых также можно выбрать свободно распространяемые ОС.

Традиционная схема обучения основам программирования выглядит следующим образом:

- понятие алгоритма с использованием алгоритмических нотаций, таких как машина Тьюринга, нормальные алгоритмы Маркова и др.
- основы базового императивного языка программирования, в качестве которого, как правило, выступают Pascal, Basic, Fortran, C.
- базовые навыки составления алгоритмов
- введение в низкоуровневое программирование на основе избранной архитектуры (чаще всего – 8086), сопровождаемое изучением соответствующего языка ассемблера
- возможно, введение в императивно-объектное программирование на основе C++, Java или Delphi

Изучение альтернативных языков программирования, таких как Lisp, Prolog и т.п., выносится на поздние стадии обучения, если вообще включается в программу.

Такой подход мы называем *монопарадигмальным*.

К достоинствам такого подхода можно отнести то, что студент получает навыки программирования на одном или нескольких популярных языках программирования.

Также следует отметить, что соответствующие среды программирования широко доступны, равно как и учебные пособия.

В то же время такая схема обладает рядом существенных недостатков. Главным из них является ограниченность чисто императивного подхода, включающая два аспекта. Во-первых, за рамками поля зрения студента остается ряд полезных методов программирования; так, многие студенты, прошедшие подготовку по традиционной схеме, не умеют пользоваться рекурсией, не умеют достаточно гибко использовать динамические структуры данных и т.п.

Во-вторых, не для всех студентов обучение на основе императивного программирования является наиболее подходящим. Ряд студентов испытывают трудности при освоении чисто императивного подхода. В рамках монопарадигмального обучения эти трудности распространяются на весь процесс программирования в целом, лишая студента перспектив освоения программирования на профессиональном уровне.

Существует альтернативная точка зрения, состоящая в том, что процесс начального обучения следует построить с одновременным использованием разнородных парадигм программирования (включая, кроме традиционной императивной, как минимум функциональную и логическую) на возможно более ранних стадиях. В этом случае обучение может включать одновременное изучение нескольких (разнородных) языков программирования. Назовем такой подход *мультипарадигмальным*.

При таком подходе подача материала строится на основе рассмотрения разнообразных приемов решения одной и той же задачи, в то время как при монопарадигмальном подходе решение (и, как следствие, мышление студента) ограничивается возможностями выбранного языка программирования.

Императивную парадигму, равно как и любую другую, нельзя считать универсальной, т.е. одинаково подходящей для любых возникающих задач. При мультипарадигмальном подходе студенты учатся оценивать достоинства и недостатки разнородных подходов, приемов и языков программирования.

Встречаются студенты, индивидуальные особенности которых располагают к изучению альтернативных парадигм программирования; в частности, некоторым студентам функциональное или логическое программирование дается легче, чем традиционное императивное. Нельзя не отметить, что переход от функционально-декларативных методов программирования к императивным существенно проще, чем в обратную сторону; студентов, ориентированных на императивное программирование, зачастую вообще невозможно переучить.

Одним из главных препятствий к внедрению мультипарадигмального подхода является отсутствие массового и общепринятого языка, сочетающего разнородные парадигмы. В частности, язык C++, часто называемый мультипарадигмальным, основан, тем не менее, на сочетании процедурного и объектно-ориентированного программирования и поддерживает лишь смежные с ними парадигмы (например, обобщенное программирование), не стимулируя при этом мышление, например, в терминах функционального программирования. Попытки создания такого языка неоднократно предпринимались, однако разработанные в результате языки широкого распространения не получили. Поэтому в процессе применения мультипарадигмального подхода приходится использовать несколько языков программирования (например, Pascal, Lisp, Prolog и Refal).

В результате особое значение приобретает выбор соответствующих систем программирования и, разумеется, операционной системы.

Многие из рассматриваемых языков программирования выглядят неестественно в

окружении, основанном на графических интерфейсах пользователя (GUI). Функциональное и логическое программирование в его чистом виде запрещает побочные эффекты функций, поэтому для взаимодействия с GUI приходится нарушать концептуальную целостность соответствующих языков.

Большинство интегрированных сред разработки основное внимание концентрируют на создании GUI, что является, безусловно, сложной, но чрезмерно специфичной задачей, которая выходит за рамки обучения основам программирования. Утверждается, что изучение основ алгоритмизации и программирования является само по себе достаточно сложной задачей, чтобы распылять внимание и силы учащихся на изучение многочисленных и несовместимых между собой сред разработки.

Поэтому мы считаем, что для обучения основам программирования в рамках мультипарадигмального подхода следует опираться на концепцию консольных приложений, которая достаточно универсальна для работы в любом языковом окружении. При этом внимание студента концентрируется не на особенностях системы программирования и средствах ввода-вывода, а на сути решаемой задачи и реализуемых алгоритмах.

Таким образом, одним из основных требований к используемому окружению является наличие развитой культуры консольных приложений.

Другим важным требованием является легкость в установке и сопровождении, а также доступность соответствующих программных средств. Так, некоторые коммерческие системы программирования следует исключить из рассмотрения по причине их дороговизны. Кроме того, задача поддержки различных проприетарных систем в условиях компьютерного класса на несколько десятков рабочих мест требует зачастую недопустимо высоких трудозатрат.

Ключевым вопросом в выборе программного обеспечения является выбор операционной системы. В отличие от языка программирования можно выбрать единую операционную систему, удовлетворяющую требованиям мультипарадигмального подхода. В настоящее время существует только две возможности: системы семейства Unix и MS Windows. Другие операционные системы (MacOS, VAX/VMS и др.) исключаются из-за слабой распространенности или дороговизны соответствующих аппаратных платформ. Популярная некогда система MS-DOS безнадежно устарела.

Большинство систем программирования под MS Windows не удовлетворяет всем перечисленным выше требованиям. Особенно следует отметить отсутствие в традициях MS Windows упоминавшейся выше культуры консольных приложений, в результате чего разработка программ на альтернативных языках программирования вызывает у студентов ложное ощущение ущербности используемых выразительных средств.

В то же время операционные системы семейства Unix предлагают широкий выбор свободных систем программирования, которые полностью удовлетворяют перечисленным требованиям (развитая культура консольных приложений, единообразная дистрибуция и администрирование, доступность). В особенности следует обратить внимание на свободно распространяемые операционные системы (*BSD и Linux). В базовую конфигурацию дистрибутивов большинства таких систем входят средства разработки для альтернативных языков программирования.

На основе всего сказанного выше можно сделать однозначный выбор в пользу применения свободно распространяемого программного обеспечения при обучении студентов программистских специальностей в рамках мультипарадигмального подхода.