1 (17), 2008

УДК 519.682.1

Прикладная математика

О НЕКОТОРЫХ АСПЕКТАХ ПРЕПОДАВАНИЯ ЯЗЫКА С++ В ВУЗАХ

С.А.Аветисян

Языки программирования входят во все типовые программы высшего образования технических направлений. Полученные в процессе обучения знания и приобретенные практические навыки должны способствовать эффективному освоению многих учебных дисциплин. Это возможно в случае, если в вузе учебный процесс поддерживается новыми педагогическими методиками преподавания на базе передовых компьютерных технологий.

Базовым языком программирования в вузах является язык С++. Не будем оспаривать правомерность выбора данного языка для преподавания программирования в вузах. Влияние на выбор языка программирования С++ оказали скорее не языковые конструкции, а технологическая необходимость использования его на предприятиях индустрии, к тому же все последующие языки программирования (java, c#) используют синтаксис С++. На сегодняшний день преподавание языка С++ в вузах не способствует обучению студента современным принципам разработки программ. Как правило, начальный курс обучения состоит из 1-го этапа (1-й семестр), основанного на обучении процедурному программированию, и 2-го этапа (2-й семестр) основанного на обучении объектно-ориентированному программированию.

В данной статье рассматриваются трудности, возникающие при преподавании данного курса и предлагаются методы обучения, которые дадут возможность студенту стать грамотным специалистом.

1.Приципы обучения 1-ого этапа начального курса (1-й семестр).

Рассмотрим, с какими трудностями сталкивается преподаватель при обучении программированию на базе языка С++ на 1-м этапе, которые возникают из-за низкого уровня обучения школьного курса информатики и стереотипа преподавания:

- У большинства студентов не бывает развито алгоритмическое мышление, которое должно прививаться при изучении курса информатики в школе.
- У студента развита привычка к стереотипу мышления, основанного на выучивании стандартных приёмов и решении только типовых задач.
- Трудности преподавания, возникающие из-за разного уровня начальных знаний по информатике.
- В вузах не уделяется должного внимания принципам разработки программ, соответствующих современным технологическим требованиям.

Эти трудности заставляют преподавателя ориентироваться на слабого студента, что снижает интерес к предмету у более сильных. Курс обучения должен быть направлен не на слабого, а на студента среднего уровня и выше. Исходя из этого преподаватель в процессе обучения не должен в основном ориентироваться на решении типовых задач. Лекции должны давать основные концепции для понимания сильных сторон языка с разбором на простых примерах. Практические занятия должны развивать самостоятельное, нестандартное мышление, современный подход к разработке программ.

Рассмотрим, как можно преодолеть перечисленные трудности.

1.1. Активность и самостоятельная работа студентов. Преподавательский опыт автора показывает, что большую роль играет самостоятельная работа студентов в интерактивной среде. Необходимо с первых же занятий использовать интерактивную среду разработки, которая обучает студента находить ошибки, думать и самостоятельно выходить из ситуаций. Только продолжительное общение с компьютером, многочисленные эксперименты и обращение к HELP-средствам позволяют овладевать непосредственно разработкой эффективных приемов и технологий.

Как правило, активность студентов зависит от понимания текущего материала, и с целью закрепления каждой темы желательно, чтобы студенты по каждой пройденной теме получали соответствующие задания. Задания, охватывающие 2 - 4 занятия, заранее раздаются студентам для подготовки и сдаются в назначенный срок в процессе обучения. При разработке заданий желательно использовать принцип индивидуального подхода. Задания должны содержать задачи разной сложности, в зависимости от уровня подготовки студента. Могут быть разработаны три степени сложности заданий. Прием и тестирование заданий желательно осуществлять интерактивной средой.

Необходим подбор интересных задач с неординарными решениями, имеющих практическое значение. Показывать разный подход в решении одной и той же задачи. Использование разнообразных и интересных по содержанию задач дает практически гораздо больший эффект, чем только лекционный курс и изучение многочисленных учебников.

Самостоятельная работа студентов, позволяет достичь ощутимых результатов в освоении данной дисциплины. Это позволит исключить трудоемкие и рутинные работы при дальнейшем обучении и к концу первого курса ликвидировать неравномерный уровень знаний студента.

1.2 Принципы разработки программ. В процессе преподавания следует исключить блок-схемный подход в разработке программ. Этот стиль алгоритмизации был правомерен до появления структурных языков. Зачастую легче написать на структурном языке программирования, нежели строить блок-схему и по нему писать программу.

Основной акцент необходимо сделать на понятии функции и разбиении программы на независимые функции, что в дальнейшем позволит плавно перейти к объектно-ориентированному программированию. На первом этапе – этапе разработки, следует научить студента представлять программу в виде иерархии функций, где описывается не то, как выполняется программа, а что надо выполнить, из каких функций она должна состоять. Функция рассматривается как черный ящик с определением входа (входные параметры) и выхода (возвращаемое значение). Используются принципы функционального проектирования, проектирования сверху-вниз.

2-й этап – непосредственное программирование, где каждая функция раскрывается посредством базовых элементов языка: итерации, рекурсии, логических утверждений и используются приемы структурного программирования. Прежде чем подключить функцию к основной программе, необходимо ее протестировать, проверить правильность ее работы. Как правило, главная программа (main) – тестирующая, осуществляет интерфейс с пользователем и вызывает обрабатывающие функции.

Студент должен научиться выделять короткие функции – обработки, ввода, вывода, диалоговые (интерфейсные) с последующим подключением в единую программу на простых, но не стандартных задачах. В процессе обучения студентом должен быть создан свой набор утилитарных функций, которые в виде отдельного файла, подключаются к программе на этапе компоновки.

Структурная запись программы играет немаловажное значение в процессе разработки и важно привить студенту на начальном этапе обучения грамотную запись программы, т.е. структурную запись команд, использование комментариев, смысловых имен переменных. Программа должна быть читабельной, понятной и должна быть самотестируемой. С самого начала надо учить составлять алгоритмы с анализом их правильности, встраивать конструкции для проверки кода.

2. Приципы обучения второго этапа начального курса (2-ой семестр).

Второй этап обучения это – освоение объектно-ориентированного программирования (ООП). Основная задача на этом этапе – научить студента объектному мышлению, т.е. мышлению категориями объектов. Зачастую легче научить мыслить объектно, нежели процедурно. Такой подход ближе к реальности, более понятен и интересен студенту,

Принципы объектно-ориентированного и структурного программирования не являются взаимно исключаюшими. Приемы структурного программирования используются во всех объектно-ориентированных конструкциях [1]. Необходимо, чтобы студент хорошо представлял отличие объектно-ориентированного программирования от процедурного программирования и осознавал необходимость перехода к объектно-ориентированному программированию. Преподаватель должен осуществить плавный переход от процедурного программирования, где базовым понятием является функция, к объектно-ориентированному программированию и понятию объекта.

Студенты обучение объектно-ориентированному программированию принимают с большим энтузиазмом, чем обучение процедурному программированию и легче воспринимают новый стиль программирования [2]. Этот стиль ближе к реальному миру, развивает способность к творчеству, логику, умение строить и использовать абстракции.

В отличие от чисто объектно-ориентированных языков программирования (Smalltalk, Java, C#), C++ является объектно-ориентированной надстройкой над процедурным языком C, имеет много сложных для понимания объектности деталей. Это затрудняет изучение объектно-ориентированной парадигмы на базе языка C++, и важно при изучении ООП научить студента проектированию классов с использованием базовых UML диаграмм. Следует научить студента проектировать классы так, чтобы они описывали действительное поведение объекта. Студент должен рассматривать программу как взаимодействие реальных объектов. Обучение должно сопровождаться подбором смысловых, интересных задач и выполнением соответствующих заданий.

На втором этапе начального курса, прежде чем переходить к непосредственному объектно-ориентированному программированию на языке C++, необходимо сначала изучить фундаментальные принципы объектно-ориентированного подхода, уделить внимание освоению объектно-ориентированного мышления, концепции упрятывания информации, модульности и проектирования интерфейсов, обосновать преимущество объектного подхода перед процедурным.

Так как многие концепции объектно-ориентированного программирования в C++ могут быть поняты только после представления процесса компиляции, компоновки, распределения памяти (конструирование объекта, виртуальные и невиртуальные функции), надо обратить внимание на необходимые детали реализации. При изучении ООП на основе языка C++ можно ограничиться тем подмножеством языка C++, которое является базовым и включено во все чисто объектно-ориентированные языки (Smalltalk, Java).

Одним из важных преимуществ объектно-ориентированных языков программирования является отслеживание и обработка исключительных ситуаций (try-catch блоки). В процессе

обучения ООП, при реализации класса, необходимо обеспечить обработку ошибок, возникающих во время выполнения программы.

Для применения в учебном процессе выше предложенной методики преподавания языка C++ необходимо следующее:

- Проводить практические занятия в компьютерных классах с наличием доски и проектора, где каждому студенту предоставляется компьютер.
- Каждый этап (семестр) должен состоять из лекционного курса (32 часа) и практических занятий не менее 64 часов.
- На первых практических занятиях студенты должны сдать тесты по быстрому вводу текста посредством интерактивных пакетов обучения.
- Должны быть составлены задания с установленными сроками сдачи в процессе обучения.
- Иметь список сложных задач для получения высокого бала, выполнение которых требуется по завершении курса.
- Студентам должны быть предоставлены компьютерные классы для самостоятельных занятий и консультаций.
- При оценивании задания необходимо уделять внимание технологическим принципам разработки, прозрачности программ, самотестируемости, наличию комментариев.

По мнению автора данные рекомендации позволят поднять уровень преподавания языка программирования C++ на начальном этапе обучения и дадут возможность студенту использовать передовые технологии разработки программ в процессе дальнейшей специализации.

Ամփոփում

Հոդվածում նկարագրված է ուսուցման սկզբնական փուլում C++ ծրագրավորման լեզվի դասավանդման մի շարք սկզբունքներ, որոնք թույլ են տալիս ուսանողին օգտագործել մասնագիտացման ընթացքում ծրագրերի մշակման արդի տեխնոլոգիաները:

Литература

1. Мэтт Вайсфельд. Объектно-ориентированный подход:Java, Net,C++.пер. с англ. М.: КУДИЦ-ОБРАЗ, 2005.
2. Н. Лсонова, Н.Коробков, Преподавание объектно - ориентированного программирования в системе лицей-вуз., МИФИ г. Москва.

ЕрГУ, кафедра программирования и информационных технологий