Ю. И. Валеева

МЕТОДИКА ПРЕПОДАВАНИЯ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ПРОГРАММИРОВАНИЯ С УЧЕТОМ РЕЙТИНГОВОГО ОЦЕНИВАНИЯ ПРОЕКТОВ СТУДЕНТОВ

Студенты, пришедшие в ВУЗ, являются продуктом сегодняшней школы. В силу целого ряда объективных и субъективных причин подготовку абитуриентов по предмету «информатика» синхронной назвать нельзя: существуют профили, где серьезно занимаются программированием (информационно-технологический, физико-технический профили), другие - изучают только пользовательский курс (гуманитарные профили).

В таких условиях обучение студентов, находящихся на разных уровнях обученности программированию, имеет свою специфику. Оптимальным решением трудности сложившейся ситуации оказалось использование модульно-рейтинговой системы оценивания. Именно такой подход в оценивании позволяет каждому студенту выбирать в изучаемом модуле те задания, которые соответствуют уровню его возможностей и притязаний.

В последнее время все большее количество вузов переходят на модульно-рейтинговую систему оценки успеваемости студентов. На практике модульно-рейтинговая система используется вузами чаще при оценивании технических дисциплин, потому что рейтинговой оценкой наиболее достоверно можно «взвесить» точные знания. В свою очередь модульно-рейтинговая система контроля складывается из двух взаимосвязанных и взаимодополняющих друг друга частей - модульную и рейтинговую, которые в совокупности функционируют с большей продуктивностью и положительно влияют на успеваемость и учебную дисциплину, повышая прочность знаний студентов.

Современная формирующаяся новая парадигма образования выдвигает в качестве одной из главных - задачу достижения реально высокой эффективности профессиональной деятельности педагогических кадров. Следовательно, необходим инструментарий измерения и оценивания уровня профессиональной компетентности при формировании ее у студентов. В условиях информатизации образования в качестве такого инструментария может выступать рейтинговое оценивание.

Как показывает практика курсы «Программирование», «Информатика и программирование», «Практикум по решению задач на ЭВМ» легко разбиваются на модули, где каждый модуль, включает в себя все виды учебной деятельности, в том числе и отчетную, поэтому студенту и преподавателю удобнее подвести итог за небольшой тематический фрагмент, чем за весь семестровый период. Семестровая же рейтинговая оценка складывается из суммы рейтинговых оценок, полученных по каждому модулю.

В ходе оценивания учебных достижений, практиковалась система оценивания решения задач, которая повышает мотивацию обучения. В зависимости от уровня сложности и как следствие трудоемкости в решении задачи необходимо учитывать следующие критерии:

- оформление интерфейса задачи в соответствии с требованиями, предъявляемыми в курсе программирования;
- рациональность и оригинальность решения;
- проверка данных, введенных пользователем;
- возможность работы программы при любых входных значениях;
- отладка и тестирование программы.

Каждая задача оценивается в баллах по следующему принципу: 1 балл - правильное оформление и грамотный комментарий, 1 балл - самостоятельная отладка и тестирование, 1 балл - оригинальность и «красота» представленного решения, остальное количество баллов оценивает уровень сложности решенной задачи. Кроме того, обучаясь в системе модульно-рейтингового оценивания, студент имеет возможность «примерить на себя» позицию педагога. Умение определять рейтинговую оценку необходимо будущему учителю для того, чтобы научиться правильно судить об уровне сложности задачи, согласно системе оценивания и требований к созданию интерфейса. Это умение студент может выработать, решая самостоятельно каждую задачу, оценивая ее и сравнивая свою оценку с предложенной примерной.

Использование визуальных сред программирования невозможно без грамотного, эстетически корректного оформления интерфейса задачи, что гарантирует соблюдение ниже перечисленных требований.

Оформление интерфейса задачи

В заголовке формы - слово «Задача» и номер задачи. Условие задачи должно находиться в многострочной метке, прижатой к верхней границе формы, шрифт крупный (читабельный), без грамматических ошибок. Кнопка «Выход» расположена в правом нижнем углу формы. Компоненты размещены на форме равномерно, в логической последовательности, при этом форма должна быть оптимального размера. Надписи корректны, точны, функциональны.

Интерфейс должен быть:

1. функциональным, т.е.:

- комментарии, надписи и всплывающие подсказки корректные и однозначные для понимания;
- легким и удобным в работе с программой;
- доступны все ключевые функции программы;
- отображены все внесенные пользователем изменения;
- оптимально соответствовать новому состоянию при видимом изменении;
- при сохранении и открытии файла автоматически добавлять расширение;
- предусмотрен выход из любой ситуации;

- корректная работа меню;
- предусмотрена справочная информация и информация об авторе.

2. грамотным, т.е. текст:

- не содержит ошибок;
- удобен для чтения, прост, краток и лаконичен;
- эмоционально нейтрален и корректен.

3. этичным:

- для визуального оформления использованы оптимальные ненавязчивые, спокойные цвета для постоянной работы, а яркие - для элементов, требующих привлечения внимания;
- гармоничное сочетание цветов;
- редкое и обоснованное использование мигающего изображения или текста.

4. организованным:

- расположение компонентов логично, т.е. объединено по смыслу или другому признаку и при этом, четко отделено друг от друга;
- распределение компонентов на форме равномерное;
- пропорциональное задание размеров компонентов;
- компоненты выровнены по высоте и длине относительно друг друга;
- применяемый стиль единый во всем приложении и согласован с операционной средой;
- информация не фрагментирована и не разбросана;
- одни и те же опции должны одинаково называться;
- появление информации по необходимости и скрывание после использования (для освобождения рабочего пространства);
- вывод критических окон сообщений всегда в центре экрана.

После правильного оформления интерфейса задачи - что представляет только часть решения, требуется правильно написать код решения, сопровождая грамотными комментариями, к написанию которых тоже предъявляются свои особые специфические требования.

Код программы должен быть:

1. надежным:

- вводимые значения корректны:
  - символы не отображаются или отображаются символом * (или любым другим), когда вводится секретная информация;
  - предусмотрены сообщения об ошибках, когда ожидалось число, а ввели сроку или вообще ничего не ввели;
  - предусмотрено очищение, когда это нужно;
  - верное отображение строк.
- система устойчива к любым ошибкам, которые может допустить пользователь, т.е. выдает сообщение об ошибках, а не выбрасывает из программы;

2. правильным:

- правильное задание начальных и конечных значений циклов, условий циклов и ветвлений;
- правильное и последовательное задание параметров подпрограмм;
- использование разных идентификаторов для локальных и глобальных переменных;
- при выделении памяти под объект, очищение ее по окончании работы с этим объектом;
- арифметические выражения не содержат логических ошибок;
- предусмотрен выход из подпрограммы;
- при учете времени должна быть возможность приостановить работу программы и продолжить с того же места;
- корректное выполнение всех действий;
- одинаковое задание одних и тех же действий и опций меню;
- отсутствие невыполнимых команд;

3. качественным:

- выходная информация должна быть полной и понятной пользователю;
- представленное решение должно иметь точное соответствие условию поставленной задачи. [3]

По стилю изложения кода существуют только некоторые общие рекомендации. Хорошо, когда программа содержит комментарии, они помещаются в фигурных скобках или после знака комментария «//» в любом месте программы.

Назначение комментарий: сделать программу более «читабельной» (понятной). Программа, особенно сложная, пишется не за один день, поэтому целесообразно ставить комментарии для групп операторов, действие которых неочевидно. Присутствие формулировки задачи в комментарии поможет быстро вспомнить назначение программы.

Различают два вида комментариев - низкого и высокого уровня.

Комментарий к программе «низкого уровня» говорит о том, что и так ясно по оператору. Комментарий же «высокого уровня» объясняет смысл оператора. Поэтому лучше небольшое число комментарий высокого уровня, чем множество низкого уровня, поясняющих очевидные вещи и делающих программу неудобочитаемой. Рейтинговый балл добавляется только в случае комментарий высокого уровня.

Хорошим стилем считается использование имен переменных, отражающих их смысл, т.е. так называемые мнемонические имена (например: min, max, S, Sum, V и т.д.).

Еще один важный фактор, влияющий на понимание программы - расположение операторов. Рекомендуется писать операторы на отдельных строках. Группу операторов вложенного блока пишут со сдвигом от начала строки, что в высокоуровневых языках позволяет устанавливать соответствие составного оператора begin и end.

Рассмотрим вышесказанное на примере решения не сложной задачи.

Задача. Дана строка. Убрать из строки все встречающиеся пары, введенных пользователем, символов.

```
// первый вариант решения, наиболее часто реализуемый
S:=Edit1.Text; // ввод строки из первого текстового окна
// ввод пары символов из второго текстового окна
// перебор символов в строке от первого до предпоследнего
i:=1;
While i<Length(S) Do Begin
        // поиск в строке и удаление искомой пары
        If Copy(S, i, 2)=C Then Begin
                Delete(S, i, 2);
                Dec(i, 2);
        End;
        Inc(i); // индекс следующего символа
End;
Edit3.Text:=S; // вывод результата в третье текстовое окно
{второй вариант решения, наиболее предпочтительный, с «красивым» решением}
// ввод исходных данных в текстовые окна
S:=Edit1.Text;
C:=Edit2.Text;
// удаление всех искомых пар, встречающихся в строке
While Pos(C, S)>0 Do
        Delete(S, Pos(C, S), 2);
```

Очевидно, что во втором случае приводится оригинальное и более элегантное решение, чем в первом. Первый вариант принесет студенту 1 балл, тогда как оптимальное и «красивое» решение задачи во втором варианте позволит получить дополнительные рейтинговые баллы.

Рассмотрим решение еще одной задачи.

Задача. Дана строка. Вывести каждое слово на отдельной строке.

Предлагаем сравнить два варианта решения поставленной задачи.

{первый вариант}

```
C:= '?., !-:;+=*/0123456789'; //возможные символы, не являющиеся буквами
// поиск и вывод слов
j:=1;
While j<=Length(S) Do
Begin
    // пропуск символов, не являющихся буквами
    While (j<=Length(S)) and (Pos(Copy(S, j, 1), C)>0) Do Inc (j);
    // составление слова из букв
    w:='';
    While (j<=Length(S)) and (Pos(Copy(S, j, 1), C)<=0 ) Do
    begin
        w:=w + Copy(S, j, 1);
        Inc(j);
    end;
    // вывод слова в многострочное текстовое окно, если оно найдено
    If w <> '' Then MmSlowa.Lines.Add(w);
end; [4]
        {второй вариант}
C:= '?., !-:;+=*/0123456789'; //возможные символы, не являющиеся буквами
// поиск и вывод слов
S:=S+' ';
j:=1;
w:='';
While j<=Length(S) Do
Begin
    // составление слова из букв и вывод в многострочное текстовое окно
    If Pos(Copy(S, j, 1), C)<=0 Then w:=w + Copy(S, j, 1)
    Else If w <> '' Then Begin
                MmSlowa.Lines.Add(w);
                w:='';
                End;
    Inc(j);
end;
```

Во втором случае использовался, так называемый, признак конца слова: текущий символ не является знаком препинанием или пробелом, а следующий за ним - знак препинания или пробел. Данный подход позволяет выделять все слова, в том числе и последнее, даже если после него нет никаких других символов. При этом уменьшается количество вложенных циклов и как следствие, решение упрощается для восприятия.

По окончании написания кода проекта важным этапом является отладка и тестирование программы, на котором выявляются и устраняются ошибки. Различают три вида ошибок:

- синтаксические;
- семантические и ошибки выполнения;
- логические.

К синтаксическим ошибкам относят ошибки, допускаемые в правилах записи тех или иных конструкций (например: отсутствие описания переменных, забыли «;», и т.п.). Такие ошибки легко устраняются, т.к. обнаруживаются компилятором.

К семантическим или ошибкам выполнения относятся ошибки, возникающие при выполнении программы из-за ошибок в самой программе, отсутствия проверки входных данных на допустимость значений или ввода некорректных данных (например: деление на ноль).

Логические ошибки - те, что не приводят к завершению выполнения программы, однако при некотором наборе данных вследствие этих ошибок может быть неверный результат. При решении задачи использован неверный алгоритм выполнения, нарушения могут быть как в ходе рассуждения, так и в написании формул. Выявляться логические ошибки могут только при грамотном тестировании. В основной своей массе, студенты не утруждают себя полноценным тестированием, и как следствие, не умеют его грамотно составлять, а значит и полноценно проверять решение задачи. Трудность в составлении тестов для проверки правильности решения задачи состоит в подборе таких входных данных, которые бы показали наличие логических ошибок и выявили семантические. На каждую задачу заранее составляются несколько тестов (от 5 и более): определенным исходным данным определяются выходные данные, включая сообщения о невозможности решения при данных входящих значениях.

На основе опыта преподавания в Кузбасской государственной педагогической академии были разработаны модули и соответствующие им рейтинговые оценки для каждого учебного элемента, модуля и общей итоговой рейтинговой оценки за курс. Разработанное учебно-методическое пособие, содержащее кроме теоретических основ, практические задачи с указанием предполагаемым количеством баллов за решение, прошло апробацию и получило гриф УМО в 2013 году.

Хочется подчеркнуть важность и педагогическую ценность использования модульно-рейтингового оценивания при преподавании объектно-ориентированного программирования, дающего возможность находить оптимальные, логически выверенные решения, осуществлять поддержку процесса обучения, побуждающего стремиться к саморазвитию, обеспечивающего формирование способности организовывать свой труд на научной основе, самостоятельно анализировать и оценивать полученные результаты, а так же формулировать необходимые выводы.

Список литературы

1. Валеева Ю.И. Объектно-ориентированное программирование в среде Delphi: Учебное пособие для студентов. - Новокузнецк: изд-во КузГПА, 2005. - 154с.
2. Масленников А.С, Шебашев В.Е. Организация учебного процесса на основе модульно-рейтинговой технологии // Фундаментальные исследования. - 2007. - № 2 - стр. 68-70 URL: www.rae.ru/fs/?section=content&op=show_article&article_id=2551 (дата обращения: 9.03.2014).
3. Можаров М.С. Практикум по решению задач в среде Lazarus с использованием модульно-рейтинговой системы оценивания. Учебно-методическое пособие для студентов. / М.С. Можаров, Ю.И. Валеева, Л.В. Попова - Новокузнецк: изд-во КузГПА, 2013. - с.150

4. Можаров М.С., Бойченко Г.Н. Основы структурного программирования: Учебное пособие / М.С. Можаров, Г.Н. Бойченко. - Новокузнецк: Изд-во КузГПА, 2006. - 220 с.
5. Попова Л.В. Проблемы реализации рейтинговой системы оценки знаний студентов. // сайт: Электронный научный журнал. - [Электронный ресурс]. URL: http://journal.kuzspa.ru/articles (дата обращения: 6.03.2014)
6. Рязанова О.Ю. Применение рейтинга в преподавании дисциплины «Техническая механика» // сайт: Социальная сеть работников образования. - [Электронный ресурс]. URL: http://nsportal.ru/npo-spo/obrazovanie-i-pedagogika/library/primenenie-reytinga-v-prepodavanii-distsipliny (дата обращения: 9.03.2014)
7. Можаров М.С., Коткин С.Д. О развитии содержательной линии «моделирование и формализация» в школьном курсе «информатика и икт»//Информатика и образование. 2010. № 4. С. 95-99.
8. Можаров М.С. Структурное программирование в примерах и решениях// Учебное пособие. Рекомендовано УМО ГОУ ВПО РГПУ им. А.И.Герцена» в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению 050200 Физико-математическое образование (Регистрационный номер рецензии №782 от 07.04.2010г.) / Новокузнецк, 2010.
9. Можаров М.С., Журавлёв С.В. Некоторые особенности преподавания программирования за рубежом //Актуальные вопросы современной науки. 2015. № 40. С. 122-132.
10. Можаров М.С. Формирование профессиональной мобильности будущего учителя информатики // Педагогическая информатика. 2009. № 3. С. 31-37.