УДК 004.588+004.023

В. В. Лаптев

МОРФОЛОГИЧЕСКИЙ СИНТЕЗ ВАРИАНТОВ ЗАДАНИЙ В ОБУЧАЮЩЕЙ СИСТЕМЕ ПО ПРОГРАММИРОВАНИЮ

В состав учебно-методических материалов обучающей системы по программированию должен входить большой набор однотипных вариантов заданий для написания программ по всей тематике изучаемого материала. При фиксированном наборе заданий система утрачивает свою обучающую функцию после выполнения пользователем всех заданий. Вследствие этого обучающая система должна обладать способностью генерировать вариант задания «на лету». Помимо тестовых заданий в обучающей системе должны присутствовать задания по написанию и изменению программного кода разных видов: написание законченной программы, написание фрагмента программы, исправление программы с ошибками, рефакторинг кода программы, написание модульного текста для заданного кода и т. п. Генерацию вариантов заданий можно осуществить на основе принципа морфологического синтеза. Типовое задание на программирование по некоторой теме должно представлять собой параметризованный шаблон с параметрами. Конкретный вариант задания формируется системой посредством выбора значений параметров. Разнообразие вариантов определяется сочетаниями параметров. На основе анализа конкретных заданий определены три вида параметров: независимые, зависимые и вложенные. Значения параметров – двух типов: множество строковых значений, вычисляемые системой числовые значения. Обучающая система при создании конкретного варианта сначала должна выбрать значения независимых параметров. Значения зависимых параметров выбираются на основании уже известных значений независимых параметров. И для каждого значения, с которым связаны вложенные параметры, система должна сгенерировать их значения. Для представления шаблона в составе базы заданий обучающей системы должен быть разработан предметно-ориентированный язык программирования, например, на основе xml. Конкретный вариант задания формируется посредством текстовой подстановки конкретного значения параметра на соответствующее место в теле шаблона аналогично тому, как это делает практически любой макропроцессор. Для генерации в заданиях арифметических выражений со стандартными функциями разработан вероятностный алгоритм, порождающий выражение по грамматике.

Ключевые слова: обучающая система, программирование, шаблон задания, параметры шаблона, вариант задания, морфологический синтез, алгоритм генерации выражений

Введение

В [1] были сформулированы требования к среде обучения программированию. В частности, формирование банка заданий – это одна из важнейших задач, которую необходимо решить разработчикам. В состав учебно-методических материалов системы должен входить большой набор однотипных вариантов заданий для написания программ по всей тематике изучаемого материала. В [2] или в [3] можно найти наборы таких заданий. С одной стороны, однотипность заданий в значительной мере упрощает процесс проверки и оценивания работ обучаемых, т. к. для каждого типа преподаватель может определить единые критерии оценивания. С другой стороны, выполнение множества однотипных упражнений позволяет обучаемому добиться достаточно прочного усвоения данной конкретной темы.

При фиксированном наборе заданий, даже достаточно большом, система утрачивает свою обучающую функцию после выполнения пользователем всех заданий, поэтому очевидно, что обучающая система должна обладать способностью генерировать вариант задания «на лету». Эта проблема практически не затрагивается в исследованиях по автоматизации обучения программированию. В [4–10] рассматривается генерация вариантов тестовых заданий для произвольной предметной области, хотя в [8, 10] приводятся примеры тестовых заданий по программированию. В [10] описаны параметризуемые тестовые задания открытого типа по программированию. Однако с помощью тестирования можно только проверить некоторые знания в области программирования, но невозможно научиться программировать.

Таким образом, при автоматизации обучения программированию возникает задача разработки *метода генерации заданий* по программированию. Генерацию вариантов можно осуществить на основе морфологического синтеза [11]. Метод морфологического синтеза предполагает,

что имеется типовое «изделие» и некоторый набор его параметров. Для каждого параметра известны множества значений. Комбинируя значения параметров, можно получать разные варианты целевого «изделия». В нашем случае типовым «изделием» является типовое задание по программированию. Разнообразие вариантов заданий определяется сочетаниями параметров.

Для решения этой проблемы необходимо:

– проанализировать виды заданий, используемых при обучении программированию;

– для каждого вида заданий определить, что является типовым заданием;

– определить набор параметров и множества значений для каждого вида задания;

– разработать алгоритмы генерации вариантов;

– разработать представление задания в составе обучающей системы.

Типовое задание на программирование по некоторой теме должно представлять собой параметризованный шаблон, значения параметров которого должны быть сформированы системой для каждого конкретного варианта. Подобный подход использовался в [10] для генерации конкретных вариантов тестовых вопросов.

Виды заданий при обучении программированию

При обучении программированию тесты с вопросами закрытого типа могут быть использованы как вспомогательный вид заданий, с помощью которых можно выявить знания по определенной тематике. Например, такие задания полезны для выявления знаний стандарта языка программирования. Обучающая система может применять тестирование как в качество входного контроля для допуска к обучающему материалу определенного уровня трудности, так и в качестве текущего контроля в процессе обучения.

Одним из видов профессиональной деятельности программиста является чтение исходных кодов программ (как своих, так и чужих), поэтому весьма полезными являются тестовые задания, в которых необходимо читать программный код (или блок-схему) и определять ход выполнения программы. Тестовые задания по программированию, рассмотренные в [8–10], можно использовать для обучения навыкам чтения программного кода. Подобные задания используются и в ЕГЭ по информатике [12].

Чтение и понимание кода необходимы программисту и при тестировании программы. Одним из обязательных видов заданий должны быть задания на составление тестов для проверки программы. Программа представлена в задании, а программист должен составить набор тестов для проверки корректности ее выполнения. Такие задания легко формулируются в тестовой форме либо в виде вопросов с выбором нескольких альтернатив, либо в виде открытых вопросов, которые требуют явного ответа пользователя. Это удивительно, но нам не встречались публикации, посвященные разработке и применению подобных заданий при обучении программированию.

Однако основной деятельностью разработчика программного обеспечения (ПО) является написание программ, поэтому набор заданий в обучающей системе должен включать задания на программирование. Эти задания выполняются в редакторе кода обучающей системы, которая может при этом работать в одном из режимов: в обучающем, тренировочном или контрольном [13]. Задания на программирование могут быть двух видов, которые мы назовем *полными* и *частичными*.

Полные задания – это задания на разработку законченной программы (и набора тестов к ней). Подобные задания представлены в ЕГЭ по информатике [12] – это задача С4. Очевидно, что этот вид заданий должен быть основным при обучении программированию.

Частичные задания не требуют написания законченного программного кода – работа осуществляется с фрагментами кода. На наш взгляд, следующие виды частичных заданий являются чрезвычайно важными при обучении программированию:

1. Задан программный код, необходимо найти и исправить ошибки.
2. Задан неполный программный код, необходимо добавить фрагмент кода в соответствии с заданием.
3. Задан программный код, требуется преобразовать код (выполнить рефакторинг [14]).
4. Задан программный код, необходимо написать модульный тест для проверки работы программы.
5. Задан программный код, требуется протестировать данную программу.

Подобные задания тоже представлены в ЕГЭ [12]: в задаче С1 требуется прочитать код программы и исправить его в соответствии с заданием; в задаче С2 необходимо разработать алгоритм и представить его либо на псевдокоде, либо как фрагмент кода на одном из языков программирования.

Частичные задания могут быть весьма полезными при изучении отдельных тем. Например, опыт преподавания программирования показывает, что начинающие программисты очень часто делают ошибки при написании арифметических выражений с делением, при написании сложных логических выражений и условий циклов.

Именно поэтому обучающая система должна «уметь» генерировать не только полные, но и подобные частичные задания.

Таким образом, резюмируя, можно определить следующие виды заданий, для которых обучающая система должна «уметь» генерировать множество вариантов:

1. Тестовые задания для выявления знаний.
2. Тестовые задания для обучения навыкам чтения и понимания программного кода.
3. Задания для обучения навыкам тестирования программного кода:

– составление программных тестов;
– написание программного кода модульного теста.

4. Задания для обучения навыкам чтения и преобразования программного кода:

– исправление ошибок;
– модификация кода (рефакторинг).

5. Задания по программированию:

– частичные;
– полные.

Обучающая система должна быть способна генерировать варианты для всех этих видов заданий. Поскольку генерация тестовых заданий достаточно хорошо исследована, мы не будем ее рассматривать (см., например, те же работы [4–10]). Нами будет рассматриваться только генерация заданий вида 5, поскольку при обучении программированию подобные задания должны составлять абсолютное большинство. Сгенерированный вариант должен представлять собой *точное техническое задание* на разработку учебной программы.

Пример

Для того чтобы понять, чем точное техническое задание отличается от типичных формулировок, приводимых в различных сборниках задач по программированию, рассмотрим фрагмент типичного задания по программированию обработки файлов на C++ [2, 3]. Можно считать этот фрагмент частичным заданием по данной теме.

Написать функцию, которая создает файл, записывая в него 100 чисел в диапазоне от –50 до +50.

Очевидно, что данная формулировка задания не является полной и точной, и студент при написании программы вынужден будет обращаться к преподавателю с вопросами для уточнения требований. В данном случае все уточнения можно условно разделить на три группы:

1. Вопросы о данных.
2. Вопросы о файле.
3. Вопросы о параметрах функции.

С данными связано, как минимум, два уточнения:

- тип записываемых в файл чисел: целые или вещественные;
- размер чисел (например, целые размером 2 байта);
- способ формирования множества чисел: генерация случайных чисел, вычисление посредством вызова стандартных функций или ввод данных вручную пользователем.

По файлу уточнений требуется несколько больше. В первую очередь требуется указать тип файла: текстовый или двоичный. Если файл двоичный, то дальнейших уточнений не требуется. Для текстового файла необходимо уточнить детали форматирования:

1. Формат вывода чисел.
2. Количество выводимых чисел в одной строке файла.

3. Если выводимых чисел в строке файла должно быть несколько, то требуется явно указать разделитель.

Заметим, что конкретный формат вывода существенно зависит от типа чисел и требует дальнейших уточнений. Например, целые числа могут быть выведены либо в десятичной системе счисления, либо в шестнадцатеричной, либо в восьмеричной. А для вещественных чисел требуется указать как форму вывода (с фиксированной точкой или экспоненциальная), так и количество выводимых знаков после запятой.

Однако наибольшее количество вопросов возникает в связи с параметрами функции. В простейшем варианте функция параметров не имеет и не возвращает никакого результата. В этом случае необходимо определить, каким способом задаются необходимые для работы данные. Например, требуется определить, как задаются границы диапазона чисел:

- определяются как локальные (или глобальные) константы в программе;
- вводятся пользователем с клавиатуры.

Аналогичные уточнения требуются и для задания количества выводимых чисел, и для задания имени выводимого файла.

Однако в зависимости от потребности (изучаемой темы и цели обучения) параметрами реализуемой функции могут быть следующие данные:

1. Количество выводимых в файл чисел.
2. Диапазон выводимых в файл чисел (два параметра).
3. Тип выводимых чисел.
4. Тип файла.
5. Имя файла (на диске).
6. Файловая переменная.

Отметим, что первые четыре параметра могут быть только входными, а имя файла и файловая переменная могут быть как входными, так и выходными. Вместо любого выходного параметра можно задать возвращаемый результат функции – это еще одно уточнение, которое требуется при написании этой функции.

Можно определить в качестве входных параметров все сведения о форматировании текстового файла: формат выводимых чисел, количество чисел в строке, разделитель. Более того, можно определить переменный список параметров, т. к. в зависимости от типа файла требуются или не требуются параметры форматирования.

С параметрами связана еще одна проблема – некоторые из них необходимо проверять на корректность. Например, количество записываемых чисел не может быть нулевым или отрицательным; при указании диапазона чисел левая граница не может быть больше правой; параметр типа файла может принимать только два значения. С учетом этого в задании должно быть указано, нужно ли при реализации функции писать операторы для проверки параметров. И, естественно, возникает вопрос, каким способом должна обрабатываться аварийная ситуация.

Таким образом, мы видим, что представленный выше типичный текст задания может (и должен) быть уточнен в очень широких пределах, поэтому формулировка задания должна выглядеть как точное техническое задание.

Шаблон задания и его параметры

Рассмотрим конкретные варианты задания по приведенному выше примеру и разберемся в проблеме генерации этих вариантов. Конкретный вариант задания может быть таким:

Написать функцию, которая создает бинарный файл, записывая в него 111 вещественных чисел двойной точности в диапазоне от +0,5 до +2,5, вычисляемых как значение функции y = sin(x) + 1,5. Параметрами реализуемой функции являются: количество и тип чисел, имя файла. Возвращаемое значение: файловая переменная. Остальные данные должны быть определены как локальные переменные и константы.

Другой конкретный вариант может выглядеть так:

Написать функцию, которая создает текстовый файл, записывая в него 123 целых коротких числа в диапазоне от +15 до +56, генерируемых датчиком случайных чисел. Числа в файле должны быть выведены по 2 в строке, разделитель – символ «;» (точка с запятой). Параметры функции: диапазон чисел. Возвращаемое значение: имя файла. Количество чисел и имя файла должны быть заданы с клавиатуры.

В обучающей системе этот текст должен быть сгенерирован из шаблона посредством подстановки конкретных значений параметров. Очевидно, что автоматически генерировать подобный связный текст – очень сложно, поэтому шаблон задания должен быть более формализован. Необходимо детально проанализировать задания, чтобы разработать формализованное представление шаблона, по которому система будет способна генерировать варианты.

Шаблон может быть представлен в системе самыми разными способами, однако при любой форме представления в шаблоне должны быть представлены следующие составляющие:

- заготовка текста задания – тело задания;
- параметры шаблона;
- набор значений или способ вычисления конкретного значения для каждого параметра;
- зависимости параметров и значений параметров.

В приведенных вариантах задания параметрами шаблона, очевидно, являются:

- тип создаваемого файла;
- количество записываемых чисел;
- тип (и размер, определяемый типом) чисел;
- диапазон чисел;
- способ вычисления выводимых чисел;
- параметры реализуемой функции;
- возвращаемое значение реализуемой функции.

Менее очевидными, но необходимыми являются параметры, позволяющие указать, каким способом должны быть определены константы при выполнении задания:

- способ определения количества выводимых чисел;
- способ определения диапазона чисел;
- способ определения имени файла;
- способы определения файловой переменной.

Рассмотрим теперь, каковы множества значений этих параметров, как они могут быть представлены в шаблоне, какие ограничения накладываются на значения параметров, каковы зависимости между параметрами.

Количество записываемых чисел – это целое число, которое система может сгенерировать с помощью датчика случайных чисел и при подстановке в тело шаблона преобразовать в строковую форму. При описании данного параметра в шаблоне требуется задать диапазон значений или, по крайней мере, правую границу диапазона, предполагая, что число является положительным (что тоже должно быть указано в шаблоне). Диапазон может быть, например, от 50 до 200, т. к. в учебном примере нет необходимости создавать огромный файл – целью является освоение способов создания файла, а не количество чисел в нем.

Однако помимо самого количества чисел в задании необходимо еще указать, каким способом это число будет задаваться при выполнении задания. Значениями этого параметра является множество строк, например:

- определить как поименованную константу;
- ввести значение с клавиатуры;
- получить значение как параметр функции.

Из этого множества система выбирает один вариант и подставляет в шаблон.

Отметим важную деталь: если было выбрано одно из первых двух значений, то при выборе вариантов для параметров реализуемой функции обучающая система не должна рассматривать количество выводимых чисел как параметр функции. Это очевидная зависимость двух параметров шаблона.

Множество значений параметра-типа выводимых чисел представляет собой множество слов-терминов. Для вещественных чисел существуют устойчивые термины: вещественные одинарной точности и вещественные двойной точности, которые представлены во всех языках программирования. Типы и размеры целых чисел, напротив, весьма разнообразны. Например, в C++ существует пять типов целых чисел (char, short, int, long, long long), которые разделяются еще на знаковые (signed) и беззнаковые (unsigned). Тип целого задает его размер, который определяется реализацией конкретного компилятора. Для целых существуют только три устойчивых

термина: короткие целые (обычно short), целые (обычно int) и длинные целые (обычно long). Вследствие этого для целых нужно сначала определить подходящие названия для всех возможных типов, которые и указать в шаблоне. Например, для чисел типа char подходит термин «маленькие», а для чисел типа long long – «сверхдлинные», поэтому конкретные значения данного параметра, представляемые в сгенерированный вариант, могут выглядеть так:

- маленькие знаковые целые;
- длинные беззнаковые целые;
- целые со знаком.

Параметр-диапазон представляет собой пару чисел, которые, как и параметр-количество, система может генерировать с помощью датчика случайных чисел. Однако для этого параметра тоже существуют некоторые ограничения. Во-первых, левая граница диапазона должна быть обязательно не больше правой. Это ограничение должно быть явно задано при описании данного параметра в шаблоне. Во-вторых, тип и значения границ зависят от параметра-типа выводимых чисел. Для вещественных, например, границы можно задавать тоже в виде вещественных чисел.

Однако более серьезная проблема состоит в том, что при генерации диапазона для выводимых целых чисел должен быть учтен размер типа. Отметим, что данное ограничение является контекстно-зависимым. Если, например, система уже выбрала для параметра-типа значение «маленькие знаковые целые», то максимально возможный диапазон составляет от –128 до +127. Этот диапазон, очевидно, должен быть задан в шаблоне и «привязан» к соответствующему значению параметра-типа выводимых чисел. Кроме того, для параметра-диапазона должна быть каким-то образом указана контекстная зависимость от параметра-типа.

Кроме того, должен быть задан способ определения диапазона чисел в реализуемой функции. И выбор этого способа влияет на параметр, определяющий состав параметров реализуемой функции.

Множество значений параметра шаблона, задающего параметры реализуемой функции (как и параметр-возвращаемое значение), должно включать первые пять параметров шаблона. Но в это множество можно добавить практически любые необходимые значения, которые не указаны явным образом в задании. Например, практически обязательно добавить значение «имя файла», можно включить и значение «файловая переменная».

Заметим, что данный параметр существенным образом зависит от параметров-способов определения констант при выполнении задания. Например, имя файла может быть введено с клавиатуры, определено как строковая константа в самой функции. Аналогично файловую переменную можно объявить непосредственно в функции – тогда при генерации варианта для данного параметра система не должна выбирать файловую переменную в качестве параметра.

Возможна и обратная зависимость: если система сначала выбрала в качестве параметров функции файловую переменную или количество выводимых функций, то она не должна рассматривать соответствующий параметр-способ определения.

Среди вариантов возвращаемого значения обязательным является значение «ничего», соответствующее ключевому слову void в C++. Помимо обязательного в множество значений можно добавить любые варианты. Примерный список значений может быть такой: файловая переменная, имя файла, количество записанных чисел, последнее записанное в файл число, и т. д., и т. п.

Важнейший параметр – способ вычисления выводимых чисел. Этих способов – бесконечное множество. Одним из стандартных является генерация случайных чисел. Этот вариант однозначно связан с типом и диапазоном генерируемых чисел – студент должен написать правильное выражение с датчиком случайных чисел. Первый пример показывает, что в качестве значения параметра можно указать произвольную функцию. Однако в этом случае диапазон значений должен соответствовать указанному выражению. Кроме того, студенту придется реализовать вычисление шага по аргументу x в рамках диапазона.

Таким образом, понятно, что обучающая система должна обрабатывать параметры шаблона в определенном порядке: сначала сгенерировать выражение для вычисления значений, затем уже выбрать тип и диапазон выводимых чисел.

И наконец, рассмотрим первый параметр – тип создаваемого файла. Поскольку файлы бывают только двух типов: текстовые и двоичные, то множество значений первого параметра включает всего два значения: текстовый и двоичный (бинарный). Отметим, что для второго значения в скобках записан синоним – его тоже можно использовать при подстановке в тело шаблона.

Но более интересным является то, что для текстового файла надо указать дополнительные параметры. Эти параметры «привязаны» к конкретному значению параметра. Назовем их вложенными параметрами. В данном случае можно определить три вложенных параметра:

– формат выводимых чисел;
– количество выводимых чисел в строке файла;
– разделитель между числами.

Формат выводимых чисел зависит от типа выводимых чисел: для вещественных он может быть задан как экспоненциальный или с фиксированной точкой. Для целых он может быть задан как десятичный или, например, шестнадцатеричный (C++).

Для количества выводимых чисел в строке нужно указать диапазон значений, например от 1 до 10. Учитывая, что в строке выводится как минимум 1 число, можно левую границу не указывать, задав только максимально возможное количество. При генерации варианта это количество может быть сгенерировано системой случайным образом.

Параметр-разделитель, очевидно, должен содержать множество символов, из которых система будет выбирать конкретное значение при генерации варианта.

Анализ примеров показывает, что в шаблоне значения параметров бывают двух типов:

– множество строковых значений, из которых обучающая система должна некоторым образом выбрать один и подставить в нужное место в теле шаблона;
– непосредственно вычисляемые системой числовые значения, которые преобразуются в строку и подставляются на место соответствующего параметра.

Кроме того, все параметры шаблона можно разделить на три вида:

1. Независимые параметры.
2. Зависимые параметры.
3. Вложенные параметры.

Обучающая система при создании конкретного варианта задания из шаблона в первую очередь должна выбрать (сгенерировать, вычислить) значения независимых параметров с учетом наложенных на эти параметры ограничений. Зависимые параметры выбираются на основании уже известных значений независимых параметров. И для каждого значения, с которым связаны вложенные параметры, система должна сгенерировать их значения.

Отметим, что зависимость параметров может быть самая разнообразная. В одних случаях выбор значения независимого параметра накладывает дополнительные ограничения на значения зависимого параметра. Примером является рассмотренная выше зависимость параметра-диапазона от параметра-типа выводимых чисел: выбор типа задает дополнительные ограничения на диапазон.

В других случаях выбор конкретного значения независимого параметра приводит к тому, что система должна игнорировать зависимый параметр. Примером подобной зависимости является зависимость параметра-способа определения имени файла от параметра, задающего параметры реализуемой функции: при выборе имени файла в качестве параметра функции система не должна рассматривать другие способы определения имени файла.

Вложенные параметры можно рассматривать как разновидность зависимых параметров: при выборе конкретного значения независимого параметра система либо рассматривает вложенный параметр, либо не рассматривает. Однако семантика вложенных параметров все же более проста по сравнению с семантикой зависимости параметров: вложенные параметры всегда привязаны к конкретному значению основного параметра и должны генерироваться при выборе этого значения. Во всех других случаях вложенные параметры не участвуют в процессе генерации.

С учетом проделанного анализа формализованный вариант первого задания может выглядеть так:

Написать функцию, которая создает файл, записывая в него числа.

– тип файла: бинарный;
– количество чисел: 111;
– тип чисел: вещественные двойной точности;

– диапазон чисел: от +0,5 до +2,5; определяется в теле функции;
– способ вычисления чисел: $y = \sin(x) + 1{,}5$;
– параметры реализуемой функции:
1) количество чисел;
2) тип чисел;
3) имя файла;
– возвращаемое значение: файловая переменная;
– файловая переменная: определяется в теле функции.

Здесь тело задания указывает суть задания, а вариант представлен как набор конкретных значений параметров. То же задание при подстановке значений непосредственно в тело задания может выглядеть так (значения параметров подчеркнуты):

Написать функцию, которая создает бинарный файл, записывая в него 111 чисел вещественных двойной точности в диапазоне от +0,5 до +2,5, вычисляемых как значение функции y = sin (x) + 1,5:

– параметры реализуемой функции:
1) количество чисел;
2) тип чисел;
3) имя файла;
– возвращаемое значение: файловая переменная;
– диапазон чисел: определяется в теле функции;
– файловая переменная: определяется в теле функции.

Тот или иной вид конкретного варианта зависит от представления шаблона в составе базы заданий в обучающей системе.

Оценка количества вариантов

Пусть в шаблоне определено n параметров и количество значений параметра p_i, где $1 < i \le n$, равно m_i. Если параметры попарно независимы, то максимальное количество вариантов, которые система может сгенерировать, задается известной формулой:

$$N = m_1 \cdot m_2 \cdot \ldots \cdot m_n.$$

Даже при относительно небольших значениях n и m_i количество сгенерированных вариантов достаточно велико.

Оценим количество вариантов заданий, которые можно сгенерировать по рассмотренным параметрам. Обозначим параметры как p_i, количество значений параметра – как m_i. Описанный выше список параметров выглядит так:

1) p_0 – тип создаваемого файла, независимый;
2) p_1 – количество записываемых в файл чисел, независимый;
3) p_2 – тип (и размер, определяемый типом) чисел, независимый;
4) p_3 – способ вычисления выводимых чисел, зависит от p_2;
5) p_4 – возвращаемое значение реализуемой функции, независимый;
6) p_5 – параметры реализуемой функции, независимый;
7) p_6 – диапазон чисел, зависит от p_2;
8) p_7 – способ определения количества выводимых чисел, зависит от p_5;
9) p_8 – способ определения диапазона чисел, зависит от p_2, p_3, p_5;
10) p_9 – способ определения имени файла, зависит от p_5;
11) p_{10} – способ определения файловой переменной, зависит от p_5.

Кроме того, имеются еще вложенные параметры для p_0 – 3 параметра. Итак, $m_0 = 2$. Примем $m_1 = 1$, т. к. в задании этот параметр представлен конкретной константой. Аналогично будем считать и $m_6 = 1$. Ориентируясь на описанные выше типы чисел в C++, имеем $m_2 = 7$. Пусть количество $m_3 = 2$ (генерация чисел с помощью датчика случайных чисел и вычисление чисел по некоторой функции). Для параметра p_4 примем $m_4 = 3$ (одно из значений – «ничего», второе – файловая переменная, третье – количество записанных чисел). Количество $m_5 = 7$ – это мы определили при анализе возможных параметров реализуемой функции. Количество значений $m_7 = m_8 = 3$, т. к. (помимо передачи в качестве параметров) эти данные можно определить либо как

константы, либо ввести с клавиатуры, либо сгенерировать датчиком случайных чисел. Соответственно $m_9 = 2$. И наконец, $m_{10} = 1$ – объявление файловой переменной локально в теле реализуемой функции (помимо того, что она может быть задана в качестве параметра функции).

Оценим количество вариантов заданий для p_0 = двоичный. Это количество меньше, чем для p_0 = текстовый, т. к. в последнем случае имеются еще 3 вложенных параметра, которые существенно добавляют вариантов. Без учета параметров реализуемой функции имеем (индекс обозначает номер параметра):

$$N_1 = m_2 \cdot m_3 \cdot m_4 = 7_2 \cdot 2_3 \cdot 3_4 = 42.$$

Учет параметров функции значительно увеличивает это количество. Так как реализуемая функция может иметь от 1 до 7 параметров, то общее количество различных вариантов списка параметров будет таким:

$$N(p_s) = \sum_{i=0}^{7} c_7^i = 2 \cdot (1 + 7 + 21 + 35) = 128.$$

Следовательно, даже без учета зависимых параметров количество вариантов равно:

$$N = N_1 \cdot N(p_5) = 42 \cdot 128 = 5376.$$

Этого вполне достаточно, чтобы обучаемый усвоил тему создания двоичных файлов, одновременно закрепляя изученный материал по функциям.

Заметим, что если вместо частичного задания студенту будет необходимо выполнить полное задание, то количество вариантов еще возрастет, поскольку вариативность значительно повышается. Например, количество записываемых в файл чисел может быть получено следующими способами:

1) определено как глобальная поименованная константа;
2) определено как локальная поименованная константа в главной функции;
3) определено как локальная поименованная константа в реализуемой функции;
4) вводится с клавиатуры в главной функции;
5) вводится с клавиатуры в реализуемой функции;
6) генерируется с помощью датчика случайных чисел в главной программе;
7) генерируется в реализуемой функции;
8) передается в главную функцию как параметр командной строки.

Кроме того, эта константа, получаемая каким-либо способом в главной функции, может быть передана в качестве параметра в реализуемую функцию – уменьшается зависимость параметра p_7 от параметра p_5 (что также увеличивает количество вариантов).

Представление шаблона и алгоритмы генерации

Для представления шаблона в базе обучающей системы на основе xml разработан предметно-ориентированный язык [15], названный нами taml. Теги taml представлены в таблице.

Теги языка представления шаблона задания

Тег	Назначение
<task> </task>	Начало и конец описания шаблона
<body> </body>	Начало и конец описания тела задания
<p*k*> </p*k*>	Начало и конец описания параметра, *k* – это номер параметра

Тег описания шаблона **<task>** имеет аргументы:

1) themes – темы, по которым может быть сгенерирован вариант задания из данного шаблона;
2) author – автор-разработчик данного шаблона.

Тело задания обрамляется тегами <body> </body>. В теле задания задается текст и в тексте в нужных местах проставляются маркеры параметров #p*k*, где *k* представляет собой номер параметра. Конкретный вариант задания формируется посредством текстовой подстановки конкретного значения параметра шаблона вместо соответствующего параметра в теле шаблона.

Тег описания параметра имеет аргумент type – тип параметра, имеющий 4 значения: текст (text), number (число), диапазон чисел (range), выражение (expression). Важный аргумент depend – зависимые параметры, которые задаются в виде списка < pk_1, pk_2, … , pk_n >. Остальные аргументы зависят от типа. В частности, для параметра text задается множество значений, а для чисел – диапазоны генерируемых значений.

Алгоритмы, осуществляющие текстовую подстановку в шаблон, хорошо известны и очень давно применяются в макропроцессорах [16, 17], поэтому приводить их нет необходимости – просто покажем аналогию. Шаблон – это макроопределение, в котором указаны параметры макроопределения (маркеры). Макровызов в данном случае отсутствует, т. к. аргументы, подставляемые вместо параметров, непосредственно выбираются или вычисляются обучающей системой. Это существенно упрощает обработку, поскольку нет необходимости разбирать макровызов и выделять аргументы.

Как показано выше, одним из параметров в генерируемом задании может быть некоторая функция, поэтому обучающая система обязана обеспечивать генерацию произвольных математических выражений для подстановки в разнообразные задания. Студентом Д. Тертышниковым был разработан алгоритм генерации произвольных арифметических выражений, который используется в обучающей среде Semantic IDE [13] для проверки знаний по написанию арифметических выражений на языке C++. На основе этого алгоритма нами был разработан алгоритм генерации математического выражения, который применяется в системе морфологического синтеза вариантов заданий.

Алгоритм порождает выражение с помощью простой порождающей грамматики с учетом количественных и вероятностных настроек. Грамматика выглядит так:

$$E = E + E \mid E - E \mid E*E \mid E/E \mid (E) \mid F(E)$$

$$F = \sin \mid \cos \mid \exp \mid \log \mid \ln \mid \mathrm{sqrt} \mid \mathrm{pow}$$

Символ F – это набор стандартных функций, входящих в математическую библиотеку практически любого языка программирования.

Вывод выражения $y = \sin(x) + 1{,}5$ в первом из приведенных выше примеров может быть выполнен так:

$$y = E \rightarrow y = E + E \rightarrow y = F(x) + E \rightarrow y = \sin(x) + E \rightarrow y = \sin(x) + 1{,}5.$$

Выражения в заданиях, приведенных выше, обычно бывают не очень сложные, с малым количеством операций и операндов.

Количественные настройки – это количество операций в генерируемом выражении, причем под операцией понимаются и скобки, и вызов функции. Обозначим это количество k. Заметим, что k фактически задает количество замен E при порождении выражения.

Нам также потребуется количество E в выражении – обозначим это число N_E.

Вероятностные настройки задают вероятность выбора альтернативы в правилах грамматики. Пусть для первого правила заданы вероятности p_{11}, p_{12}, p_{13}, p_{14}, p_{15}, p_{16}, причем $\sum p_{1i} = 1$. Аналогично для второго правила заданы вероятности p_{21}, p_{22}, p_{23}, p_{24}, p_{25}, p_{26}, причем $\sum p_{2i} = 1$. Заметим, что некоторые вероятности могут быть равны 0. Это означает, что данная альтернатива никогда не выбирается при генерации выражения. Например, если нам нужно сгенерировать выражение без скобок, то надо установить $p_{15} = 0$.

Идея вероятностного выбора состоит в том, что с помощью установленных значений интервал [0, 1] вещественных чисел разбивается на части, каждая из которых по длине равна соответствующей вероятности. Обозначим левые границы подинтервалов символом L, а правые – символом R. Таким образом, границы подинтервалов вычисляются так:

$[L_1, R_1)$: $L_1 = 0$; $R_1 = p_1$,
$[L_2, R_2)$: $L_2 = R_1$; $R_2 = p_1 + p_2$,
$[L_3, R_3)$: $L_3 = R_2$; $R_3 = p_1 + p_2 + p_3$,
$[L_4, R_4)$: $L_4 = R_3$; $R_4 = p_1 + p_2 + p_3 + p_4$,
$[L_5, R_5)$: $L_5 = R_4$; $R_5 = p_1 + p_2 + p_3 + p_4 + p_5$,
$[L_6, R_6]$: $L_6 = R_5$; $R_6 = p_1 + p_2 + p_3 + p_4 + p_5 + p_6 = 1$.

Для выбора альтернативы генерируется случайное число из диапазона [0, 1] и определяется, в какой из подинтервалов оно попадает – эта альтернатива и выбирается.

Тогда алгоритм генерации выглядит так:

1. Установить $n_{op} = 0$.
2. Установить вероятности выбора альтернатив и разбить интервал [0, 1] на части.
3. Установить стартовое выражение $y = E$.
4. Установить $N_E = 1$.
5. Пока $n_{op} < k$:
   - 5.1. Генерировать случайное число $0 \le d \le 1$.
   - 5.2. Определить i-й подинтервал $[L_i, R_i)$.
   - 5.3. Выбрать альтернативу E_i.
   - 5.4. Выбрать случайный номер $1 \le n \le N_E$.
   - 5.5. Заменить E_n на альтернативу E_i.
   - 5.6. Вычислить N_E.
   - 5.7. $n_{op} = n_{op} + 1$.
6. Конец цикла.

После этого необходимо «пробежаться» по сгенерированному выражению, заменяя все оставшиеся E на генерируемые числовые константы, а все F заменить на конкретные функции по той же вероятностной схеме.

Заключение

Морфологический синтез вариантов заданий в обучающей системе имеет следующие достоинства: преподавателю позволяет решить проблему подготовки однотипных заданий для студентов и облегчает проверку выполнения заданий; студенту решение множества однотипных заданий помогает добиться прочного усвоения данной конкретной темы.

СПИСОК ЛИТЕРАТУРЫ

1. *Лаптев В. В.* Требования к современной обучающей среде по программированию / В. В. Лаптев // Объектные системы-2010 (Зимняя сессия): материалы II Междунар. науч.-практ. конф., Россия, Ростов-на-Дону, 10–12 ноября 2010 г. Ростов н/Д, 2010. С. 104–110.

2. *Павловская Т. А.* С/С++. Программирование на языке высокого уровня / Т. А. Павловская. СПб.: Питер, 2004. 461 с.

3. *Лаптев В. В.* В. С++. Объектно-ориентированное программирование. Задачи и упражнения / В. В. Лаптев, А. В. Морозов, А. В. Бокова. СПб.: Питер, 2007. 288 с.

4. *Башмаков А. И.* Разработка компьютерных учебников и обучающих систем / А. И. Башмаков, И. А. Башмаков. М.: Информ.-изд. дом «Филинъ», 2003. 616 с.

5. *Кручинин В. В.* Генераторы в компьютерных учебных программах / В. В. Кручинин. Томск: Изд-во Томск. ун-та, 2003. 203 с.

6. *Посов И. А.* Автоматизация процесса разработки и использования многовариантных учебных заданий: автореф. дис. … канд. техн. наук / И. А. Посов. СПб.: СПбГУ, 2012. 18 с.

7. *Морозова Ю. В.* Компьютерная поддержка самостоятельной работы студентов на основе генераторов тестовых заданий: автореф. дис. … канд. техн. наук / Ю. В. Морозова. Новосибирск: ТУСУР, 2011. 20 с.

8. *Кручинин В. В.* Методы и алгоритмы построения компьютерных учебных программ и систем на основе генерации информационных объектов: дис. … д-ра техн. наук / В. В. Кручинин. Томск: ТУСУР, 2005. 413 с.

9. *Кручинин В. В.* Модели и алгоритмы генерации задач в компьютерном тестировании / В. В. Кручинин, Ю. В. Морозова // Изв. Томск. политехн. ун-та. 2004. Т. 307, № 5. С. 127–131.

10. *Sosnovsky S.* Web-based Parameterized Question as a Tool for Learning / S. Sosnovsky, O. Shcherbinina, P. Brusilovsky // In Allison Rossett (ed.): Proceedings of E-Learn 2003, Phoenix, Arizona USA, November 7–11, 2003, p. 2151–2154 // URL: http://www.pitt.edu/~peterb/papers/ELearn03.pdf.

11. *Андрейчиков А. В.* Системный анализ и синтез стратегических решений в инноватике: концептуальное проектирование инновационных систем / А. В. Андрейчиков, О. Н. Андрейчикова. М.: ЛЕНАНД, 2014. 432 с.

12. *Сафонов И. К.* ЕГЭнциклопедия. Информатика / И. К. Сафонов. СПб.: БХВ-Петербург, 2010. 496 с.

13. *Грачев Д. А.* Семантическая интегрированная среда для обучения программированию / Д. А. Грачев, В. В. Лаптев // Педагогическая информатика. 2013. № 2. С. 71–81.

14. *Фаулер М.* Рефакторинг: улучшение существующего кода / М. Фаулер. СПб.: Символ-Плюс, 2003. 432 с.

15. *Фаулер М.* Предметно-ориентированные языки программирования / М. Фаулер. М.: Вильямс, 2011. 576 с.

16. *Кемпбелл-Келли М.* Введение в макросы / М. Кемпбелл-Келли. М.: Сов. радио, 1978. 152 с.

17. *Браун П.* Макропроцессоры и мобильность программного обеспечения / П. Браун. М.: Мир, 1977. 256 с.

Статья поступила в редакцию 22.12.2014,
в окончательном варианте – 12.01.2015

ИНФОРМАЦИЯ ОБ АВТОРЕ

Лаптев Валерий Викторович – Россия, 414056, Астрахань; Астраханский государственный технический университет; канд. техн. наук, доцент; доцент кафедры «Автоматизированные системы обработки информации и управления»; Laptev@ilabsltd.com.

V. V. Laptev

MORPHOLOGICAL SYNTHESIS OF EXERCISES IN THE TRAINING SYSTEM OF PROGRAMMING

Abstract. A part of teaching materials training system programming should include a large set of similar variants of tasks for writing programs for all the themes of the studied material. For a fixed set of tasks, the system loses its learning function as soon as the users executes all the tasks Therefore, the training system must be able to generate an option of the tasks "on the fly". In addition to the test tasks, the training system must comprise tasks for writing and editing code of different types: writing complete programs, writing a part of the program, fixing an error, refactoring a program code, writing a module text for the given code, etc. The generation of the variants of the tasks can be performed on the basis of the morphological synthesis. A typical task on programming on a certain subject should be a parameterized template with parameters. A specific version of the task is formed by the system by means of selecting parameter values. The combinations of the parameters determine the variety of the options. Based on the analysis of the specific tasks, three categories of the parameters: independent, dependent and nested are identified. There are two types of the values of the parameters: a set of string values and numeric values calculated by the system. When creating a particular option, the training system must first choose the values of the independent parameters. The values of the dependent parameters are chosen on the basis of the known values of independent parameters. And, for each value, which is associated with the nested parameters, the system must generate their values. To view the template in the base tasks the training system should have a domain specific language, such as xml-based. A specific task is shaped by textual substitution of the specific parameter value to the appropriate place in the body of the template in the same way as almost any macro-processor does. To generate arithmetic expressions with standard features in tasks, a probabilistic algorithm that generates a grammar expression is developed.

Key words: training system, programming, exercise template, template settings, type of task, morphological synthesis, algorithm of expression generation.

REFERENCES

1. Laptev V. V. Trebovaniia k sovremennoi obuchaiushchei srede po programmirovaniiu [Requirements to the modern training media on programming]. *Ob"ektnye sistemy-2010 (Zimniaia sessiia). Materialy II Mezhdunarodnoi nauchno-prakticheskoi konferentsii. Rossiia, Rostov-na-Donu, 10–12 noiabria 2010 g.* Rostov-on-Don, 2010. P. 104–110.

2. Pavlovskaia T. A. *S/S++. Programmirovanie na iazyke vysokogo urovnia* [C/C++. Highly qualified programming]. Saint Petersburg, Piter Publ., 2004. 461 p.

3. Laptev V. V., Morozov A. V., Bokova A. V. S++. *Ob"ektno-orientirovannoe programmirovanie. Zadachi i uprazhneniia* [C++. Object-oriented programming. Tasks and exercises]. Saint Petersburg, Piter Publ., 2007. 288 p.

4. Bashmakov A. I., Bashmakov I. A. *Razrabotka komp'iuternykh uchebnikov i obuchaiushchikh sistem* [Development of computer textbooks and training systems]. Moscow, Informatsionno-izdatel'skii dom «Filin», 2003. 616 p.

5. Kruchinin V. V. *Generatory v komp'iuternykh uchebnykh programmakh* [Generators in computer training programs]. Tomsk, Izd-vo Tomskogo universiteta, 2003. 203 p.

6. Posov I. A. *Avtomatizatsiia protsessa razrabotki i ispol'zovaniia mnogovariantnykh uchebnykh zadanii. Avtoref. dis. kand. tekhn. nauk* [Automation of the development and use of multivariant training tasks. Abstract of dis. cand. tech. sci.]. Saint Petersburg, SPbGU, 2012. 18 p.

7. Morozova Iu. V. *Komp'iuternaia podderzhka samostoiatel'noi raboty studentov na osnove generatorov testovykh zadanii. Avtoref. dis. kand. tekhn. nauk* [Computer support of the independent student work on the basis of generators of test tasks. Abstract of dis. cand. tech. sci.]. Novosibirsk, TUSUR, 2011. 20 p.

8. Kruchinin V. V. *Metody i algoritmy postroeniia komp'iuternykh uchebnykh programm i sistem na osnove generatsii informatsionnykh ob"ektov*. Dissertatsiia dokt. tekhn. nauk [Methods and algorithms of development of the computer training programs and systems based on generation of information units. Dis. doc. tech. sci.]. Tomsk, TUSUR, 2005. 413 p.

9. Kruchinin V. V., Morozova Iu. V. Modeli i algoritmy generatsii zadach v komp'iuternom testirovanii [Models and algorithms of task generation in computer testing]. *Izvestiia Tomskogo politekhnicheskogo universiteta*, 2004, vol. 307, no. 5, pp. 127–131.

10. Sosnovsky S., Shcherbinina O., Brusilovsky P. Web-based Parameterized Question as a Tool for Learning. In Allison Rossett (ed.): *Proceedings of E-Learn 2003, Phoenix, Arizona USA, November 7–11, 2003*, pp. 2151–2154. Available at: http://www.pitt.edu/~peterb/papers/ELearn03.pdf.

11. Andreichikov A. V., Andreichikova O. N. *Sistemnyi analiz i sintez strategicheskikh reshenii v innovatike: kontseptual'noe proektirovanie innovatsionnykh sistem* [System analysis and synthesis of strategic decisions: conceptual designing of innovation systems]. Moscow, LENAND Publ., 2014. 432 p.

12. Safonov I. K. *EGEntsiklopediia. Informatika* [United public encyclopedia. Computer science]. Saint Petersburg, BKhV-Peterburg, 2010. 496 p.

13. Grachev D. A., Laptev V. V. Semanticheskaia integrirovannaia sreda dlia obucheniia programmirovaniiu [Semantic integrated medium for programming training]. *Pedagogicheskaia informatika*, 2013, no. 2, pp. 71–81.

14. Fauler M. Refaktoring: uluchshenie sushchestvuiushchego koda [Refactoring: improvement of the existing code]. Saint Petersburg, Simvol-Plius Publ., 2003. 432 p.

15. Fauler M. *Predmetno-orientirovannye iazyki programmirovaniia* [Subject-oriented programming languages]. Moscow, Vil'iams Publ., 2011. 576 p.

16. Kempbell-Kelli M. *Vvedenie v makrosy* [Introduction to macros]. Moscow, Sovetskoe radio Publ., 1978. 152 p.

17. Braun P. *Makroprotsessory i mobil'nost' programmnogo obespecheniia* [Macroprocessors and mobility of software]. Moscow, Mir Publ., 1977. 256 p.

The article submitted to the editors 22.12.2014,
in the final version – 12.01.2015

INFORMATION ABOUT THE AUTHOR

Laptev Valeriy Victorovich – Russia, 414056, Astrakhan; Astrakhan State Technical University; Candidate of Technical Sciences, Assistant Professor; Assistant Professor of the Department "Automated Systems of Information Processing and Management"; Laptev@ilabsltd.com.