-программа 2 выделения только трехсвязных графов из массива полученных программой 1 графов;

-программа 3, которая в гамильтонов трехсвязный простой граф, полученный на этапе 2, вносит две новые вершины: одна из которых лежит на гамильтоновом контуре, а вторая находится вне его.

Обоснованием введения программы 3 являются следующее предположение:

Предположение 1. Все однородные графы с числом вершин n и степенью 3, не имеющие мостов, имеют гамильтонов контур либо контур максимальной длины из n-1 вершин.

Примером может служить граф Петерсена, либо упомянутые выше три однородных графа степени 3.

Все полученные на этапе 3 графы проверяются программой 4 на наличие гамильтонового контура. В результате графы, не имеющие гамильтонового контура, составляют искомое множество.

Данный комплекс программ позволит вычислить долю трехсвязных плоских графов в общем числе плоских однородных графов степени 3, а также установить долю таких графов в общем числе однородных кубических графов. Отметим, что последнее возможно выполнить только для однородных кубических графов с $n < 24$, т.к. число таких графов велико и последнее рассчитанное значение выполнено для $n = 24$ [4]. Согласно [4] время расчета составило около 10 часов.

В данном проекте также реализуется 2 вида генерации графов: путём послойного наращивания количества вершин в плоских трёхсвязных графах, а также путём склейки плоских трёхсвязных графов (K_4^3, K_6^3, K_8^3,..) и различных перестановок. В ходе исследований планируется определение возможности генерации новых, ранее неизвестных плоских трёхсвязных графов, а также более эффективного способа их генерации, который способен охватить большее множество генерируемых графов и за меньшее время работы.

Литература

1. Харари, Ф. Теория графов / Ф. Харари –М.: Мир, 1973. – С. 87.
2. Емиличев, В.А. Лекции по теории графов / В.А. Емиличев, О.И. Мельников, В.И. Сорванов, Р.И. Тышкевич – Москва: Наука, 1990. – С. 203.
3. Grunbaum, B. Contex polytopes, Wiley, New York, 1967. – P. 359.
4. Meringer, M. Erzeugung regularer graphen.Bayreuth,1996. – S. 36.

УДК 681.3

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБУЧЕНИИ ПРОГРАММИРОВАНИЮ

Брич А.Л.
УО «Брестский государственный технический университет», г. Брест

Обучение программированию предполагает выработку у обучаемых специальных навыков работы с компьютером, включая разработку алгоритмов решаемых задач, кодирование их на языках программирования. Для обучения разработке алгоритмов, написанию и отладке программ наряду с обучающими системами в настоящее время преимущественно используются существующие системы программирования.

При этом обучение алгоритмизации, тесно связанное с оценкой корректности алгоритмов и поиском ошибок, как правило, производится вручную. Качество обучения алгоритмизации, его эффективность может быть в значительной мере повышена использованием компьютерных сред, обеспечивающих исполнимость алгоритмов по аналогии с системами программирования.

Важная часть такой среды - входной язык, лингвистические средства. Так текст разрабатываемого алгоритма должен быть инвариантен к языкам программирования высокого уровня, читаемым, легко контролируемым вручную и автоматически в процессе разработки и в итоге должен быть готовым документом для кодирования на подходящем языке программирования.

В работе, которая является продолжением и развитием [1, 2], приводятся результаты апробации принципов построения входного языка и соответствующих инструментальных средств. Лингвистическое обеспечение такой системы должно быть построено на принципах структурного программирования. Кроме этого, получаемый текст должен обладать исполнимостью. Входной язык, соответствуя базовым принципам построения языков программирования, должен основываться на минимальном наборе изобразительных средств, управляющих структур, отображающих соответствующие математические понятия и правила пользования ими.

Используется процедурный механизм построения алгоритмов

<программа> ::= <основной_блок> <подпрограммы> ,

<подпрограмма> ::= <функция> | <процедура> .

Для повышения читаемости разрабатываемых текстов, в частности, должны использоваться: - комментарии, “встраиваемые” в текст без выделения специальными служебными символами, наряду с традиционными строчными комментариями; - гибкая и удобная в использовании система образования идентификаторов с применением “встраиваемых” комментариев; - “подсветка синтаксиса” текста алгоритма при работе в редакторе для повышения читаемости текста, его восприятия.

Указанные "встраиваемые" комментарии должны записываться прописными буквами кириллицы, что позволит склонять используемые в тексте алгоритма идентификаторы в зависимости от контекста, и соответственно формулировать осмысленные тексты (совокупности команд) по правилам, близким к правилам русского языка. Соответственно

<идентификатор> ::= [<приставка>] <корень> [<окончание>] { ["_" [<приставка>] <корень> [<окончание>]] } .

Здесь

<корень> ::= <прописная_буква> { [<прописная_буква> | "_" | <цифра>] } ,

<приставка> ::= <встраиваемый_комментарий> ,

<окончание> ::= <встраиваемый_комментарий> ,

<встраиваемый_комментарий> ::= <строчная_буква> { [<строчная_буква >] } .

Дополнительно в язык должны быть введены средства описания глобальных и локальных объектов, в том числе, байтового, символьного, логического и файлового типов. Указанный перечень является достаточным для разработки и описания класса типовых алгоритмов.

Набор операторов языка должен отличаться единым стилем описания и должен содержать операторы следующих типов: <оператор_присва-ивания>, <оператор_вызова>, <условный_ оператор>, <оператор_цикла_ с_предусловием>, <оператор_цикла_с_постусловием>, <оператор_ цикла_с_параметром>. При редактировании алгоритма должна обеспечиваться работа с блоками текста, буфером обмена. Соответственно инструментальные средства должны обеспечивать: редактирование алгоритмов; исполнение алгоритмов, включая контроль корректности текстов алгоритмов, трансляцию и получение исполнимого кода, управление его выполнением; пошаговое выполнение алгоритмов в целях обучения и отладки; генерацию программ на языках высокого уровня.

Работа системы макетировалась с использованием языков высокого уровня (Pascal, Delphi). Система Windows-ориентирована. Среда пользователя - интегрированная, многооконная, русифицированная. Получаемый в результате разработки текст алгоритма позволяет автоматически генерировать код (функционально-адекватное внутреннее представление алгоритма), исполняемый, например, внутренней виртуальной машиной, которая и реализует интерфейс взаимодействия пользователя с алгоритмом в процессе его разработки и отладки.

Литература

1. Муравьев, Г.Л. Входной язык автоматизированной системы обучения (АСО) алгоритмизации // Новые информационные технологии в образовании: материалы 3 международной конференции NITE' 98, Минск, 1998 г.: в 3 т. / БГЭУ. – Минск, 1998. - Т.2. - С. 83–88.

2. Муравьев, Г.Л. Автоматизация обучения алгоритмизации и программированию / Г.Л. Муравьев, С.В. Мухов // Вести ИСЗ. - 2000. – № 3. - С. 24-29.

УДК 004.6

ОБ ОРГАНИЗАЦИИ СИСТЕМЫ ТЕСТИРОВАНИЯ

Ванюков С.В.

УО «Брестский государственный университет им. А.С.Пушкина», г. Брест

Научный руководитель – Силаев Н.В., доцент

Начиная разработку системы тестирования TENMA, была поставлена задача создать, по возможности, универсальный инструмент, который можно будет легко расширить и приспособить для любой системы тестирования. В качестве инструмента разработки был выбран язык C# на платформе Visual Studio .NET 2.0. Предоставляемые ею средства для сетевого программирования, построения классов и шифрования передаваемых данных оказались подходящими для решения поставленной перед нами задачи.

Основное внимание при проектировании уделяется открытости системы и простоте её обновления и инсталляции. Разрабатываемая система позволит: строить древовидно организованные структуры связей, моделирующие диалог естественного общения при опросе; обеспечить поддержку импорта тестов формата TQF, который применяется в настоящее время в системе теоретического тестирования на математическом факультете БрГУ.

Пакет состоит из двух частей — клиентской и серверной. Клиентская часть инсталлируется на все компьютеры, за которыми производят тестирование. Серверная часть устанавливается только на один – компьютер администрирования.

Общие требования, выдвигаемые нами к системе, таковы:

Удобство администрирования.

Совместимость. Система должна поддерживать уже существующие форматы тестов или поставляться с конвертером.

Полнота. Всё необходимое для работы системы должно быть включено в её дистрибутив, чтобы до предела упростить процесс инсталляции.

Универсальность. Система должна поддерживать различные типы вопросов и ответов. Например, вопрос может содержать иллюстрации и текстовое форматирование, ответ должен может быть только одним из предложенных, набором из нескольких предложенных вариантов или вводиться с клавиатуры.