методы предотвращения нежелательных ситуаций. Преждевременное изучение материала по предотвращению от нежелательных ситуаций, а так же подготовка к ним, могут повысить дальнейшее эффективность работы систем.

Литература

1. Обзор существующих систем автоматизации учета оптовых и розничных продаж. [Электронный ресурс]. URL: https://novainfo.ru/article/10924 (дата обращения: 28.01.2020).
2. Обзор АИС. [Электронный ресурс]. URL: https://cinref.ru/razdel/04650raznoe/09/397253.htm (дата обращения: 03.02.2020).
3. Сравнительный анализ информационных систем для автоматизации управленческого учёта [Электронный ресурс]. URL: https://sibac.info/journal/student/57/136612 (дата обращения: 07.02.2020).

А.О. Кузовенков

СОВЕРШЕНСТВОВАНИЕ ПРОЦЕССА АВТОМАТИЧЕСКОЙ ПРОВЕРКИ ЗАДАЧ ПО ПРОГРАММИРОВАНИЮ

(Тольяттинский государственный университет)

Система автоматической проверки (САП) задач по программированию – это система, которая оценивает программные решения учащихся. Какой же должна быть САП? Прежде чем ответить на данный вопрос, сначала выделим основные критерии, на основании которых необходимо производить оценку программы учащегося (см. табл. 1).

Теперь рассмотрим, как в соответствии с каждым критерием работают современные САП (см. табл. 2). Из таблицы 2 видно, что для оценки программы, существующие САП выполняют практически одинаковые действия.

Оценку корректности работы программы все САП осуществляют при помощи набора тестов, где каждый тест включает в себя входные данные и эталонный ответ, которые могут быть представлены строками или файлами. Процесс формирования тестов возможен двумя способами:

1. С помощью программы динамического генерирования тестовых данных. Используя описанный администратором алгоритм, такая программа автоматически генерирует входные и выходные данные в соответствии с условиями задания.
2. Вручную. В данном случае администратор самостоятельно создаёт набор тестов.

Вне зависимости от способа формирования тестов, необходимо чтобы полученные тесты покрывали все возможные пути выполнения тестируемой программы.

Таблица 1. Критерии оценки программы учащегося

Критерий	Описание
Оригинальность исходного кода программы	Это процентный показатель, демонстрирующий объем исходного кода программы, написанного самостоятельно. Не исключено, что некоторые студенты (например, те, которые плохо усваивают материал), чтобы выполнить задание, будут использовать в качестве своего решения чужой исходный код. Однако исходный код, как и реферат, школьное сочинение и т. п. каждый человек пишет по-своему.
Безопасность программы	Это показатель, который может принимать значения «0», если программа представляет опасность для САП, или «1», если присланная программа безопасна. Под безопасностью программы понимается безопасность её исходного кода и процесса выполнения. Иногда умышленно или нет исходный код программы может содержать опасные вставки, способные нарушить работу САП или даже полностью её остановить. Так, например, исходный код может содержать ассемблерные включения, которые способны вызывать потенциально опасные функции для работы с системой на низком уровне. Допустим, что в исходном коде нет опасных вставок, однако вывести из строя САП могут даже разрешённые команды. Например, умышленно или нет студент может прислать программу, которая во время выполнения войдёт в бесконечный цикл и зависнет или же, работая с файловой системой, удалит важные для САП файлы и каталоги.
Корректность исходного кода программы	Это некий балл, который показывает, на сколько исходный код программы соответствует требованиям, указанным в задании. Довольно часто присланная на проверку программа может выдавать правильные выходные данные, но при этом иметь исходный код, несоответствующий заданию. Значение данного критерия зависит также от результатов компиляции (или проверки интерпретатором) исходного кода программы (время компиляции, наличие предупреждений и т. д.).
Корректность работы программы	Это некий балл, который показывает, на сколько правильные данные выдаёт программа. Также на этот балл оказывает влияние количество ресурсов, которые использует программа во время выполнения.

Само тестирование программы проводится следующим образом. Вначале проверяемой программе на вход подаются входные данные теста. Затем производится сравнение выходных данных проверяемой программы с эталонным ответом теста (посимвольное сравнение двух строк с пробелами) [1]. Если вывод проверяемой программы совпадает с эталонным ответом, это означает, что тест пройден успешно. Описанный алгоритм повторяется для всех тестов набора. В конце тестирования программы подводятся итоги: определяется, какие тесты программа прошла, а какие нет и в зависимости от этого учащемуся начисляется определённое количество баллов.

Таблица 2. Результаты анализа современных САП

САП	Корректность исходного кода программы	Оригинальность исходного кода программы	Корректность работы программы	Безопасность программы
PC^2	код проверяется посредством его компиляции (или интерпретатором)	отсутствует	проверка осуществляется при помощи набора тестов	отсутствует
Contester	код проверяется посредством его компиляции (или интерпретатором)	отсутствует	проверка осуществляется при помощи набора тестов; контролируется время работы и количество использованной памяти	контролируются запрещённые библиотеки и подключаемые модули; программа выполняется в защищенном изолированном окружении
T-BMSTU	код проверяется посредством его компиляции (или интерпретатором)	используется сервер обнаружения некорректных заимствований, который выполняет интеграцию системы с внешним программным обеспечением, которое ведет поиск некорректных заимствований	проверка осуществляется при помощи набора тестов; контролируется время работы и количество использованной памяти	выполняется в защищенном изолированном окружении; запущенное решение не имеет доступа ни к сетевым интерфейсам, ни к файловой системе сервера; решение, написанное на небезопасном с точки зрения работы с памятью языке, запускается в отладчике valgrind
Ejudge	используется инструмент, проверяющий стиль кода; в остальном код проверяется посредством его компиляции (или интерпретатором)	простое сравнение содержимого двух файлов (diff)	проверка осуществляется при помощи набора тестов; контролируется время работы и количество использованной памяти	используются интегрированные системы изолированного запуска приложений и контроля над ними (песочницы)
DOMjudge	код проверяется посредством его компиляции (или интерпретатором)	простое сравнение содержимого двух файлов (diff)	проверка осуществляется при помощи набора тестов	устанавливается минимальное окружение (chroot) и ограничиваются ресурсы (cgroup)

Таким образом, на основании проведённого анализа, у существующих САП можно выделить следующий ряд недостатков:

1. Оценка исходного кода программы студента в основном ограничивается лишь результатами компиляции (или проверки интерпретатором).
2. Проверка оригинальности исходного кода программы практически не осуществляется (в основном проводится простое сравнение двух текстов).
3. Так как для проверки вывода программы используется посимвольное сравнение с эталонным ответом, чтобы покрыть все возможные пути выполнения программы, необходимо создавать довольно большое количество тестов.
4. Отсутствие комплексной проверки выполнения программы. Например, студентам дано задание написать хеш-функцию. Как известно хеш-функция – это функция, осуществляющая преобразование массива входных данных произвольной длины в (выходную) битовую строку установленной длины. Для проверки нам неважно, что именно будет выводить проверяемая программа, необходимо проверять решение на наличие коллизий. Чтобы это проверить, необходимо сравнивать пары ввод-вывод программы между собой. В существующих САП такое сравнение невозможно.

Чтобы устранить данные недостатки, были приняты следующие решения:

1. Перед тем как скомпилировать (или проверить интерпретатором) присланную программу, необходимо сначала проверить размер файла с исходным кодом. Это делается для того, чтобы отсеять слишком большие решения. Так, например, ученик может просчитать очень трудоёмкое решение на своём компьютере и поместить результаты в один большой словарь в исходном коде. А само решение будет выглядеть как read(a), write(result[a]). Таким образом, проверив размер исходного кода, можно будет сразу отклонить подобные решения. Далее нужно проверить исходный код на наличие ассемблерных вставок. Это можно сделать с помощью программы noasm. Такая проверка позволит избежать компиляции и последующего выполнения потенциально опасного кода. Затем необходимо проверить стиль исходного кода. Такая проверка нужна по причине того, что в реальных проектах очень часто с одним и тем же исходным кодом работает сразу несколько человек. Поэтому очень важно чтобы программист придерживался единого стиля. Это облегчит чтение, поддержку и слияние кода для остальных программистов. В конце должны осуществляться проверки, которые непосредственно связаны с условиями задачи. Например, условиями задачи запрещено использовать функцию sin. Проверив исходный код можно будет определить, использовал ли студент данную функцию или нет.
2. Чтобы оценить решение студента, его исходный код в обязательном порядке должен пройти через проверку на плагиат. Для определения процента оригинальности присланного исходного кода предполагается использовать ступенчатый метод [2].

3. Чтобы уменьшить количество создаваемых тестов, предполагается использование регулярных выражений. Одним регулярным выражением можно будет описать сразу несколько схожих эталонных ответов.

4. Если после прохождения основного тестирования по заданию требуется дополнительная проверка пар ввод-вывод, планируется полученные пары подавать на вход программам, которые написаны самим автором задачи. Результатом работы каждой такой программы будет процент соответствия пар ввод-вывод условиям задачи.

Таким образом, станет возможным создание системы автоматической проверки задач по программированию, которая сможет объективно оценить решения студентов.

Литература

1. Bennouar D. An Automatic Grading System Based on Dynamic Corpora [Электронный ресурс] // The International Arab Journal of Information Technology. 2017. Vol. 14.4A PP. 552-564. URL: https://pdfs.semanticscholar.org/f188/a42968741ca733178701766d1eebb9f0a410.pdf (дата обращения: 11.01.2020).

2. Стрельчёнок Г. В. Ступенчатый метод проверки исходного кода программы на плагиат [Электронный ресурс]. – 2016. – Режим доступа: https://nauchkor.ru/pubs/stupenchatyy-metod-proverki-ishodnogo-koda-programmy-na-plagiat-587d36555f1be77c40d58cf6 (дата обращения: 11.01.2020).

Ю.В. Кузьмина, В.В. Ворошилов

ОПРЕДЕЛЕНИЕ ПСИХОЛОГИЧЕСКОГО ПРОФИЛЯ МОЛОДЫХ ИССЛЕДОВАТЕЛЕЙ С ПОМОЩЬЮ ИС

(Самарский государственный технический университет)

Для лучшего взаимодействия молодых людей с научным руководителем, а также для лучшего понимания самого себя необходимо осознание отдельных личностных психологических характеристик. Они помогут понять, какие качества необходимо развивать в себе для достижения наибольшего результата.

В настоящее время улучшение качества образования в вузах является приоритетной задачей. Психологическое сопровождение студентов в период обучения, является ключевым фактором в повышении качества образования.

Однако во время выполнения научно-исследовательских работ студенты сталкиваются с множеством факторов, которые негативно сказываются на образовательном процессе и на развитии личности в целом, поэтому возникает потребность диагностики психологического состояния студентов.

Сегодня многие организации, проводящие психологические мониторинги групп лиц, используют для анализа и хранения полученной информации либо