Федеральное государственное учреждение «Федеральный научный центр
Научно-исследовательский институт системных исследований
Российской академии наук»
(ФГУ ФНЦ НИИСИ РАН)

ТРУДЫ НИИСИ РАН

ТОМ 12 № 3

МАТЕМАТИЧЕСКОЕ И КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ СЛОЖНЫХ СИСТЕМ:

ТЕОРЕТИЧЕСКИЕ И ПРИКЛАДНЫЕ АСПЕКТЫ

МОСКВА
2022

Тематика номера:
Информационные технологии в учебной информатике, информационные и компьютерные технологии, математические исследования

Журнал публикует оригинальные статьи по следующим областям исследований: математика, математическое и компьютерное моделирование, обработка изображений, визуализация, системный анализ, методы обработки сигналов, информационная безопасность, информационные технологии, высокопроизводительные вычисления, оптико-нейронные технологии, микро- и наноэлектроника, математические исследования и вопросы численного анализа, история науки и техники.

The topic of the issue:

Information technology in educational informatics, information and computer technologies, mathematical issues

The Journal publishes novel articles on the following research arias: mathematics, mathematical and computer modeling, image processing, visualization, system analysis, signal processing, information security, information technologies, high-performance computing, optical-neural technologies, micro- and nanoelectronics, mathematical issues and problems of numerical analysis, history of science and of technique.

Заведующий редакцией: В.Е.Текунов

Издатель: ФГУ ФНЦ НИИСИ РАН,
117218, Москва, Нахимовский проспект 36, к. 1

СОДЕРЖАНИЕ

CONTENT

Анализ методов проверки кода программ на плагиат в цифровой образовательной платформе Мирера

М.С. Дьяченко[1], В.А. Домрина[2], А.Г. Леонов[3], К.А. Мащенко[4], И.Г. Райко[5], А.А. Холькина[6]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, Mdyachenko@niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, domrina@niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kirill010399@vip.niisi.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ilya.rayko@niisi.ru;

[6]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kholkina@niisi.ru

Аннотация. Возможность копировать части кода программ участниками образовательных учебных курсов по программированию затрудняет объективное оценивание уровня знаний каждого обучающегося. В статье излагается вариант решения этой проблемы на примере авторской цифровой образовательной платформы Мирера. Поставлены требования к антиплагиат-анализу с учетом специфики цифровой образовательной платформы Мирера. Рассмотрены известные алгоритмы и методы для реализации антиплагиат-анализа, выделены их достоинства и недостатки. Реализован метод отпечатков на токенизированной программе, описаны примеры работы антиплагиат-анализатора. Отмечается возможность использования антиплагиат-анализа как инструмента для контролирования уровня освоения курса студентами.

Ключевые слова: цифровая образовательная платформа, цифровая образовательная платформа Мирера, антиплагиат, антиплагиат-анализ программного кода, метод отпечатков.

1. Введение

С развитием цифровых технологий образовательных процессов растет число попыток заимствования (плагиата) [8], в том числе в курсах по программированию при составлении студенских программ. При этом учащиеся копируют не столько алгоритмы решения, сколько фрагменты кода программы или даже код целиком, поэтому проблема обнаружения плагиата актуальна практически для любого курса, который сдержит задачи по написанию программ для школ и университетов [1,2,5]. Выделяют несколько уровней списывания [10, 15], от полного копирования до изменения текста программы до неузнаваемости, при этом даже примитивные методы изменения кода, такие как перестановка компонент программы, добавление пробелов и других разделителей, а также замена всех имен переменных и функций способны «обмануть» преподавателя, проверяющего решения на плагиат вручную. Стоит также отметить, что решения, находящиеся в свободном доступе [7, 11, 12], подразумевают загрузку в систему или проверку с компьютера преподавателя, но проверка на заимствование необходима и для платформ с автоматическими проверками решений, так, Codeforces и Яндекс Контест используют проприетарный антиплагиат-алгоритм [16]. В связи с этим возникает необходимость автоматической быстрой проверки всех отправляемых решений, прошедших проверку, на схожесть, которая могла бы предоставить подробный отчет при обнаружении совпадений, включающий данные списывающего, возможного автора оригинального решения и вероятность плагиата, и работала для решения, отправленного в любое время.

Такие аспекты, как необходимость учитывать стиль кода студента, разные объемы решений, язык программирования, используемый в решении и оперативное получение обратной связи вне зависимости от нагрузки на сервер, хранящий решения, формируют особые требования к программе антиплагиат-анализа. Также важной задачей является выявление и акцентирование внимания преподавателя на случаи, когда код совпадает полностью с точностью до изменений, не влияющих на алгоритм программы. В таком случае вероятность плагиата приближается к 100%, то есть списывание с очень высокой вероятностью произошло). Для уменьшения технически и психологической нагрузки преподавателя, важно также минимизировать число ситуаций, когда антиплагиат-анализатор отмечает как

списанные задания, которые на самом деле были выполнены самостоятельно. Все вышеперечисленное усложняет задачу антиплагиат-анализа и требует рассмотрения различных подходов к реализации, позволяющих добиться достоверных результатов в широком круге заданий. Некоторые из методов, с точки зрения авторов представляющие наибольший интерес, разобраны в настоящей статье вместе с анализом их достоинств и недостатков.

Следует отметить, что цифровая образовательная платформа (ЦОП) Мирера обладает некоторой внутренней спецификой, которая может как усложнить реализацию совместимых с Мирерой алгоритмов антиплагиат-анализа, проверки на заимствование, так и позволить преподавателям и составителям курсов получить более подробную и точную статистику как о студентах, замеченных в списывании решений, и примененных ими способами скрыть факт списывания, так и о стилях оформления программ студентами, работающими без заимствований. Так, хранящаяся в системе информация о количестве попыток решения задачи и времени отправки решения может послужить решающим аргументом в подтверждении гипотезы копирования или его отсутствия, а возможность анализа каждого решения предоставляет большие объемы информации о каждом студенте индивидуально. При этом метод анализа на плагиат, совместимый с платформой, должен учитывать а) наличие шаблонов – частей кода программы, предоставленных преподавателем, которые очевидно совпадают у всех студентов, и б) возможность многофайлового представления программы и наличие большого количества вариантов такого представления. Метод отпечатков на токенизированной программе, учитывающий все особенности ЦОП Мирера, встроен в платформу. В статье рассматриваются результаты реализации этого метода при различных видах копирования и списывания, а также описаны планируемые улучшения.

2. Постановка задач исследования и анализ алгоритмов

Предлагается рассмотреть требования к антиплагиату, возможные подходы к реализации и степень удовлетворения этим требованиям. Цель исследования – выбрать алгоритм для внедрения в ЦОП Мирера по результатам этой оценки.

2.1. Формулировка требований

Для формулировки требований необходимо понимать, как списывающий ученик меняет оригинальную программу и сколько времени занимают эти изменения. По объему проделанной работы списывание можно разделить на типы [9, 13, 15]. Например, студент может, получив решение, отправить его без изменений или за несколько минут поменять названия переменных, функций, заменить строчные буквы на прописные и наоборот, убрать или добавить комментарии, пробелы и другие служебные символы – добавить «шум» [3], что точно не помешает программе пройти проверку. Оба случая антиплагиат должен определять с точностью 100%, причем первый случай должен выделять особо, поскольку в большинстве случаев полное совпадение свидетельствует о списывании. Больше времени занимает перестановка местами функций, строк, циклов, объявлений переменных и других элементов программы, а также изменение условий в циклах таких, как замена while на for, $i > 9$ на $i >= 10$ в случае целого i, и разбиение одной функции на несколько и наоборот, поскольку необдуманные изменения такого типа могут нарушить логику воплощенного алгоритма. Несмотря на то, что подобного рода совпадения возникают даже в самостоятельно выполненных работах, ожидается, что антиплагиат с большой точностью определит процент совпадения. Важно учитывать то, что добавление не относящегося к алгоритму кода, например, объявление не используемых переменных или циклов, не влияющих на задачу, не должно уменьшить возвращаемую вероятность, то есть необходимо анализировать вхождение сравниваемых файлов друг в друга.

Одной из ключевых особенностей ЦОП Мирера являются механизмы поддержки шаблонов – кодов и их фрагментов, написанных преподавателем. Преподаватель может задать неудаляемость и неизменяемость элементов шаблона и Мирера обеспечит выполнение этого требования. Таким образом, элементы шаблона студент не сможет убрать из программы или изменить и от антиплагиата-анализатора требуется учитывать наличие таких шаблонов в расчете процента совпадения и определении порога совпадения – минимального процента совпадения программ, при котором считается, что списывание произошло. Порог совпадения также должен зависеть от размера программы и, при желании, регулироваться преподавателем. Также антиплагиат-анализатор должен обрабатывать решения задачи на разных языках программирования, размещенные в нескольких файлах с различными расширениями, оперативно предоставлять обратную связь, безошибочно или с высокой вероятностью определять автора списанного кода и оповещать преподавателя о подозрительных решениях, показывая, какие части программ схожи.

Как было сказано выше, важное требование к антиплагиату – работать на доступных данных и

предоставлять результаты работы в виде, поддерживаемом ЦОП Мирера, однако, не все элементы ограничивают алгоритм антиплагиат-анализа. Так, режим строгого контроля, запрещающий студенту копировать и вставлять код, замедлит возможное списывание и позволит преподавателю при необходимости отслеживать процесс решения по черновикам студентов, возможность фиксировать копирование позволит выводить предупреждение «было скопировано … строчек в файл …», свидетельствующее о почти гарантированном списывании при высоком проценте совпадения, а возможность зафиксировать время отправки решения в систему позволяет не только определить автора списанного решения, но и с большей вероятностью определить, списана работа или нет. Например, многие копирования с небольшими изменениями (перестановка строк, замена имен переменных или отсутствие изменений) происходят под конец контеста – набора задач для занятий (семинаров, домашней или контрольной работы), и, если программа не прошла проверку на антиплагиат и сильно отличается от других попыток студента, скорее всего, он списал, предположив, что не успеет поправить свое решение. Однако, если разница между отправкой двух решений небольшая, в зависимости от задачи можно заключить, что, несмотря на процент выше порога, списывания не было, ведь студент бы физически не успел столько поправить.

2.2. Общие шаги алгоритма антиплагиат-анализа при внедрении в ЦОП Мирера

Перед тем, как оценить алгоритмы и их степень соответствия вышесказанным требованиям, рассмотрим общие детали при реализации и критерии, которые могут автоматически выполняться благодаря такому подходу. Каждый алгоритм будет получать на вход полный текст программы и преобразовывать каждую строку к стандартному виду: убирать все комментарии и пробелы, заменять все прописные буквы на строчные, заменять имена переменных на V и имена функций на F (последнее слегка видоизменено для алгоритмов, привязанным к переменным (или функциям) и синтаксису вокруг них). Степень совпадения, и, соответственно, вероятность списывания, для двух программ будем рассчитывать следующим образом: из двух вероятностей, каждая из которых равна количеству совпадающих объектов (для каждого алгоритма эти объекты свои), деленному на количество всех объектов первой и второй программы соответственно и умноженному на 100%, выбираем наибольшее значение. Таким образом, обеспечена независимость от «шума» и добавления не играющих роли в алгоритме элементов, например, при добавлении лишней строки с определением переменной или цикла и переименовании всех переменных алгоритм получит вероятность 100%. О работе в реальном времени, определении автора и информации для преподавателей будет сказано позже, в разделе про реализацию метода отпечатков на токенизированной программе.

Все рассматриваемые алгоритмы удовлетворяют самому первому критерию – безошибочно определять полностью совпадающие, в том числе с точностью до «шума», программы. Следовательно, остаются следующие требования: поддержка большого количества языков, возможность работы с несколькими файлами, поддержка функциональности шаблонов, независимость от изменения порядка функций, строк, циклов и т.д. в программе, возможность передачи результатов таким образом, что по ним легко предоставить информацию преподавателю, о которой было сказано выше. Также от алгоритма требуется не иметь false positive ответов – самостоятельно выполненных решений, которые антиплагиат отметил заимствованными.

2.3. Алгоритмы и их степень соответствия поставленным требованиям

Приведем обзор алгоритмов, используемых в настоящем исследовании.

1. Побайтовое сравнение. Алгоритм собирает строки стандартного вида в одну и кодирует, сравнивая каждый байт первой программы с каждым байтом второй программы по порядку. Этот метод подходит для нахождения идентичных с точностью до «шума» программ, но чувствителен к перестановке элементов программы. Его модификации, например, сравнивание только определенных байтов, не представляют интерес, поскольку сравниваются слишком малые части программы, что приводит к неэффективности по времени работы или завышению процента совпадения (в каждом решении одной задачи с большой вероятностью есть одинаковые небольшие конструкции, однако о заимствовании это не свидетельствует). Таким образом, этот метод прост в реализации, но не удовлетворяет важным критериям.

2. Метод отпечатков для n-грамм и токенов, алгоритм просеивания.

Разобьем программу на n-граммы (последовательность n идущих подряд символов) или токены (минимальная текстовая единица), рассмотрим последовательность их хешей. Определим окно размера w как последовательность из w хешей, и будем рассматривать окна размеров t-k+1, где t - размер для гарантированного подтверждения совпадения, а n-граммы при $n < k$ не рассматриваются. t и k – параметры, которые

подбираются на соответствующем наборе данных, от их выбора зависит степень чувствительности к перестановке элементов программы и вероятность пропустить небольшие совпадения. Количество отпечатков [2],[3] также играет ключевую роль в алгоритме, будем выбирать хотя бы один отпечаток в каждом окне, чтобы ограничить максимальное расстояние между отпечатками. Алгоритм состоит в том, чтобы выбирать как отпечаток в каждом окне элемент с минимальным хешем, поскольку эти элементы с большой вероятностью одинаковы в соседних окнах. [3, 14]

Следует обратить внимание на то, что метод отпечатков не зависит от синтаксической логики программы, поэтому может использоваться в программах с шаблонами любого размера и вида и несколькими файлами, которые можно объединить в один. Алгоритм также независим от перестановки элементов, хорошо работает с заменами условий циклов и разбиением части программы на несколько [3,5]. Несмотря на наличие минусов, о которых будет сказано в следующем разделе, грамотная реализация способна использовать их для более качественной проверки в некоторых случаях или снизить негативное влияние до минимума.

Вышеописанный метод для n-грамм лежит в основе системы MOSS (Measure Of Software Similarity) [5,6], разработанной в 1997 году, которая используется многими американскими университетами как основное средство проверки, в том числе благодаря достаточно быстрому и качественному результату и отсутствию отмеченных как плагиат честно выполненных работ.

3. Метод отпечатков на токенизированной программе на основе переменных.

Метод похож на предыдущий, однако учитывает соседние к токену переменной токены и информация о них и становится отпечатком [4]. Поскольку метод опирается на синтаксическую составляющую программы, он обязан оценивать программу только с шаблоном и часто не может оценить один шаблон для справедливой оценки совпадения кода. Более того, исследования показывают меньшую эффективность, чем похожие методы [4].

4. Метод отпечатков на синтаксическом дереве на основе переменных.

Использование синтаксического дерева приводит к более сложным в реализации, но одним из самых точных методов для антиплагиата. Например, существует подход, аналогичный методу на основе переменных на токенизированной программе: на основе кода создается синтаксическое дерево, в котором выделяются вершины переменных, и рассматриваются вершины выше вершины данной переменной, содержащие больше одного потомка и с учетом информации о них создаются отпечатки [4]. Таким образом, достигается полная независимость от способа объявления и использования переменной. Метод способен предоставить более подробную и наглядную информацию о логике и синтаксической составляющей программы, однако для оценивания необходим полный код программы, и получить информацию о совпадении частей, не входящих в шаблон, не получится. Таким образом, этот метод, несомненно, имеет огромные плюсы, но несовместим с ключевыми элементами ЦОП Мирера.

Ввиду вышеперечисленных плюсов, успешном использовании в одном из самых известных антиплагиатов, а также наличии в общем доступе библиотек Python по токенизации [6] и коду основы алгоритма в общем доступе [3] было принято решение ввести в ЦОП Мирера метод отпечатков на токенизированной программе.

3. Реализация метода отпечатков на токенизированной программе

Рассмотрим детали реализации антиплагиата, его поведение в различных типах ситуаций, механизм работы в реальном времени и результаты подбора порога совпадения и параметров по итогам тестирования антиплагиата на датасете из массива решений студентов ЦОП Мирера.

3.1. Реализация алгоритма

За основу была взята библиотека copydetect [7], позволяющая сравнить несколько файлов с заданными параметрами методом, аналогичным выбранному. Данная библиотека была модифицирована для лучшей совместимости с реализованными алгоритмами работы с решениями в ЦОП Мирера.

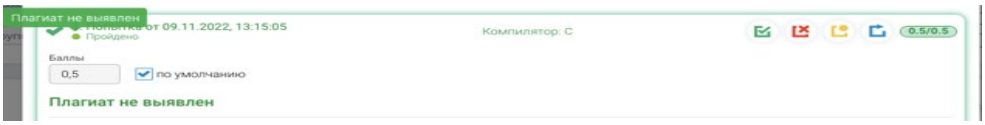

Рис. 1. Плагиат не найден

Рис. 2. Результаты проверки на антиплагиат в случае обнаружения заимствования

Так, чтобы обеспечить работу в реальном времени и обнаружить автора оригинального кода, антиплагиату передается id ученика, задачи, попытки и время отправки правильного решения вместе с кодом, при этом каждое новое решение сравнивается с уже зафиксированными для данной задачи; если список решений с процентом совпадения больше порога пуст, заключается, что заимствования не было (рис. 1), иначе автором считается ученик с самым большим процентом совпадения из этого списка (рис. 2), который по окончанию проверки преподаватель увидит в статистике с процентами схожести, временем отправления, по которому сортируются строки, и возможностью просмотреть код обеих сравниваемых программ с подчеркнутыми совпадающими строками – если в строке есть совпавшие символы, она подсвечивается желтым, если совпадает полностью с точностью до «шума» – красным (рис. 3). В статистике курса в зависимости от вероятности, возвращенной антиплагиатом, каждой задаче присваивается кружок соответствующего цвета – зеленого, если плагиат не обнаружен, желтого, если вероятность от 65% до 80%, и красного, если вероятность до 100% (рис. 4).

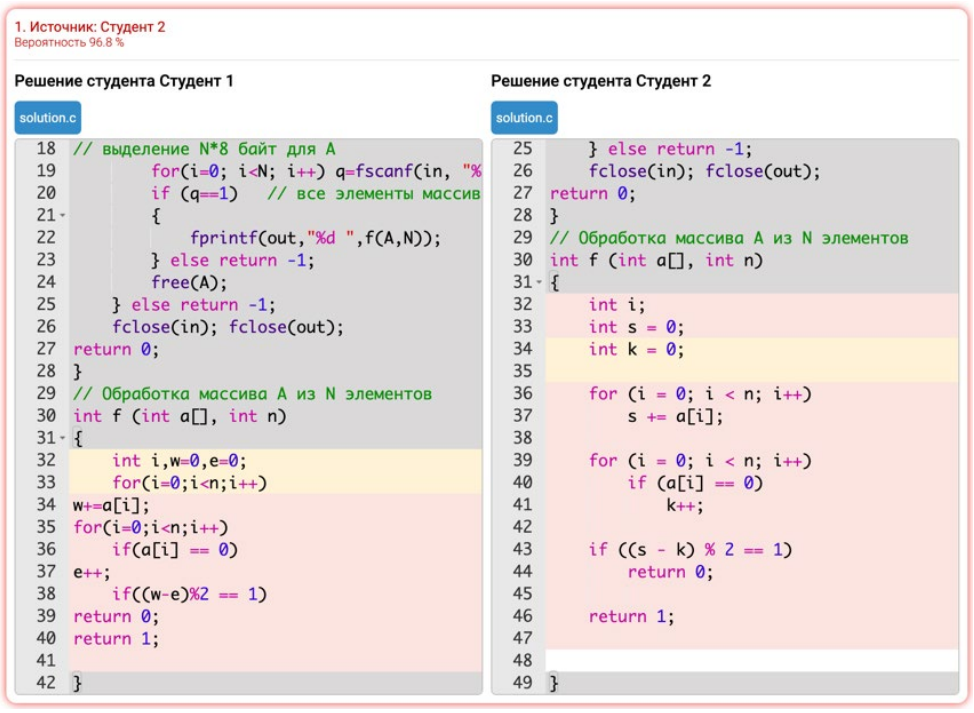

Рис. 3. Подсветка совпадающих элементов решения

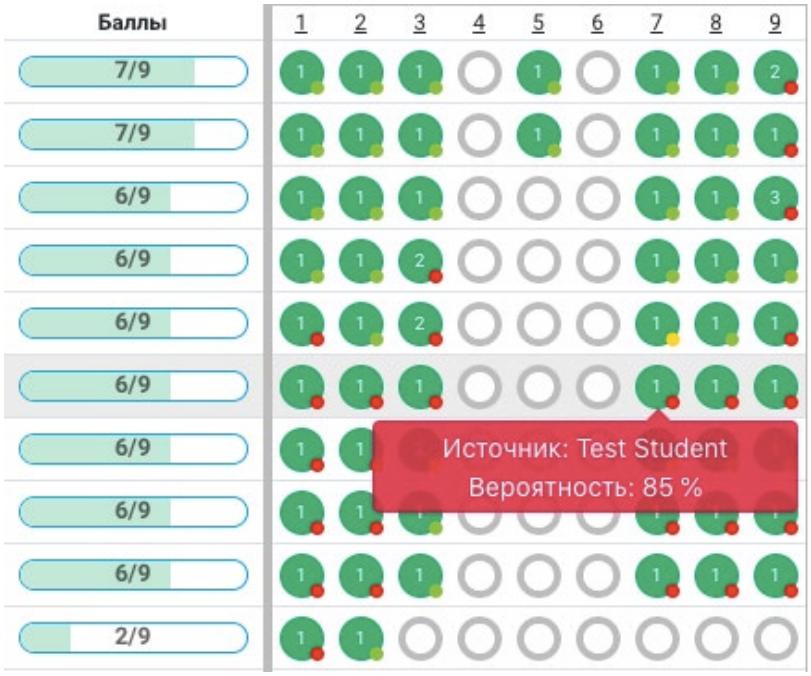

Рис. 4. Статистика решений

Особенности базы данных ЦОП Мирера позволяют сравнивать решения с решениями групп прошлых лет, у которых также было это задание, что убирает возможность незаметно получить у старшекурсников готовый код.

3.2. Подробности использования и применение антиплагиата в работе ЦОП Мирера

Поскольку метод достаточно чувствителен к сильным изменениям стиля кода, таким, как определение переменных одним оператором или разными, и использованию нескольких функций вместо одной, который обычно не меняется в попытках одного ученика, с помощью проверки всех решений студента с учетом времени между попытками при подозрении на списывание антиплагиат позволяет отследить, не было ли в какой-то момент копирования у ученика с отличающимся стилем кода или алгоритмом. Также отсутствие анализа логики кода позволяет проверять любые попытки, даже синтаксически неправильные или вызывающие ошибки компиляции, без противоречий и влияния на результат, а невозможность предоставить преподавателю подробный отчет о логике копирования и работы программы компенсируется простотой реализации и временем на получение результата. Так, даже недостатки метода могут позволить преподавателю лучше понять ситуацию в определенных случаях.

Таким образом, чаще всего антиплагиат применяется для сравнения всех правильных попыток в процессе отправки и в конце контеста для получения полной картины выполнения заданий, а при подозрении на списывание – для более тщательной проверки, в том числе не обязательно правильных попыток, и проверки попыток одного студента на сохранение стиля кода.

3.3. Проверка антиплагиата на реальных данных

Из задач определенных типов, размеров решений и шаблонов было выделено вручную несколько оригинальных и списанных решений разных типов списывания. Результаты проверки антиплагиатом с лучшим подбором параметров можно видеть в таблице ниже, где показан средний процент, было выбрано 20 задач с плагиатами следующих типов:

1. Полностью скопированное решение
2. Добавление шума (пробелы, комментарии, служебные символы)

3. Изменение названий переменных, функций и т.д.

4. Изменение условий в if с использованием отрицания и т.д.

5. Изменение порядка инициализаций переменных и функций, строк

6. Изменение циклов while и for друг на друга

7. Вынесение и внесение в функции строк, инициализаций переменных и т.д.

Таблица 1. Результаты проверки антиплагиатом с лучшим подбором параметров

	1	2	3	4	5	6	7
1	-	100%	100%	96%	100%	91%	84%
2	-	-	100%	96%	100%	91%	84%
3	-	-	-	96%	100%	89%	84%
4	-	-	-	-	96%	88%	81%
5	-	-	-	-	-	91%	82%
6	-	-	-	-	-	-	76%
7	-	-	-	-	-	-	-

По результатам видно, что типы 1-3 и 5 обнаружены с точностью 100%, 4, 6 и 7 с высокой точностью. В будущем планируется тестирование на более объемных выборках.

4. Заключение

Антиплагиат-анализ позволяет преподавателю не только оценить сходство двух отправленных решений быстрее, удобнее и точнее, чем ручная проверка, но и лучше оценить нарушения типичного стиля кода для списавшего ученика, определить момент, в который произошло списывание, указать предполагаемых авторов оригинальных решений и закономерности, по которым может происходить копирование, узнать, какие задачи вызывают наибольшую сложность и какие конструкции каких размеров чаще всего списывают студенты. Вся эта информация, несомненно, поможет преподавателю не только пресекать попытки списывания и объективно оценивать знания своих учеников, но и улучшить методику преподавания, понимая, какие задания относятся к наиболее сложным и какой материал вызывает трудности в усвоении, что поможет скорректировать курс и повысить объективность результатов проверки знаний.

В дальнейшем планируется активно апробировать антиплагиат-анализатор системы Мирера на большей выборке, включающей в себя курсы прошлых лет, поработать над наглядностью вывода результатов преподавателю, в том числе введением предупреждений вида «до конца контеста менее n минут от времени отправки решения, было скопировано более m строк, разница во времени отправки попытки менее l минут», где n, m, l могут изменяться преподавателем и отображения в статистике контестов и курсов, а также внедрить в ЦОП Мирера антиплагиат-алгоритмы, работающие с тестами и заданиями со свободным вводом, и модифицированный метод, использующий синтаксическое дерево, с учетом шаблонов.

Anti-Plagiarism Tool for Code in the Digital Educational Platform of Mirera

M.S. Diachenko, V.A.Domrina, A.G. Leonov, K.A. Mashchenko, Ilya G. Raiko, A.A. Kholkina

Abstract. The possible copying of parts of code in the solution by participants in educational programming courses makes it difficult to objectively estimate the level of knowledge of each student. The article describes a solution to this problem using the example of the author's digital educational platform Mirera. The requirements for anti-plagia-

rism tool which take the specifics of the digital educational platform Mirera into account are set. Algorithms and methods for anti-plagiarism are considered, their advantages and disadvantages are highlighted. The details of the implementation of the fingerprinting method on a tokenized program and examples of the work of anti-plagiarism tool are described. The possibility of using the tool as an instrument for better understanding the students and the degree of mastering the course is noted.

Keywords: digital educational platform, digital educational environment, Mirera, anti-plagiarism tool, tokenization, fingerprinting.

Литература

1. И. А. Посов, В. Е. Допира. Методы поиска плагиата в кодах программ. Известия СПбГЭТУ «ЛЭТИ» 2019; (6): 61-66 https://izv.etu.ru/assets/files/izvestiya-6_2019_p061-066.pdf

2. Анализ алгоритмов выявления плагиата в кодах программ, написанных на языках высокого уровня [Электронный ресурс] / Д. А. Чепрасов; Национальный исследовательский Томский политехнический университет (ТПУ) ; науч. рук. Ю. Я. Кацман // Современные техника и технологии сборник трудов XVIII международной научно-практической конференции студентов, аспирантов и молодых ученых, Томск, 9-13 апреля 2012 г: в 3 т.: / Национальный исследовательский Томский политехнический университет (ТПУ) . — 2012 . — Т. 2 . — [С. 431-432]
https://www.lib.tpu.ru/fulltext/v/Conferences/2012/C2/V2/v2_212.pdf

3. A. Aiken, S. Schleimer, D. Wikerson. Winnowing: local algorithms for document fingerprinting. In Proceedings of ACMSIGMOD Int. Conference on Management of Data, San Diego,CA, June 9–12, pp. 76–85.ACMPress, New York, USA, 2003. https://theory.stanford.edu/~aiken/publications/papers/sigmod03.pdf

4. Nick Moone. Comparing variable fingerprints to indicate plagiarism in code, June 18, 2021 https://scripties.uba.uva.nl/download?fid=682144

5. Kevin W. Bowyer and Lawrence O. Hall. Experience Using ”MOSS” to Detect Cheating On Programming Assignments. https://www3.nd.edu/~kwb/nsf-ufe/1110.pdf

Сайт библиотеки «Pygments» [Электронный ресурс]. https://pygments.org/ (дата обращения: 01.09.2022)

Библиотека «copydetect» [Электронный ресурс]. https://github.com/blingenf/copydetect (дата обращения: 01.09.2022)

6. Никитов А.В., Орчаков О.А., Чехович Ю.В. Плагиат в работах студентов и аспирантов: проблема и методы противодействия. Университетское управление: практика и анализ. 2012;(5):61-68.
https://www.umj.ru/jour/article/download/506/507

7. Perkins, Mike & Basar Gezgin, Ulas & Gordon, Raymond. (2019). Plagiarism in higher education: classification, causes and controls. Pan-Pacific Management Science, Vol. 2, 3-21 https://www.researchgate.net/publication/354143709_Plagiarism_in_higher_education_classification_causes_and_controls

8. Sraka, Dejan and Branko Kaucic. “Source code plagiarism.” Proceedings of the ITI 2009 31st International Conference on Information Technology Interfaces (2009): 461-466. https://www.semanticscholar.org/paper/Source-code-plagiarism-Sraka-Kaucic/ac10405ce5a9421b2e2dd33836350f259614931a

9. Сайт сервиса «MOSS» [Электронный ресурс]. https://theory.stanford.edu/~aiken/moss/ (дата обращения: 01.09.2022)

10. Сайт сервиса «Copyleaks» [Электронный ресурс]. https://copyleaks.com/ru/проверка-кода-на-плагиат/ (дата обращения: 01.09.2022)

11. T. Vrbanec, A. Meštrović: Taxonomy of academic plagiarism methods, Zbornik Veleučilišta u Rijeci, Vol. 9 (2021), No. 1, pp. 283-300

12. Яшин Гліб Євгенович, Хмелюк Марина Сергіївна Метод просеивания для выявления плагиата в программном коде на языке C# // Перший Незалежний Науковий Вісник. 2015. №1-1. URL: https://cyberleninka.ru/article/n/metod-proseivaniya-dlya-vyyavleniya-plagiata-v-programmnom-kode-na-yazyke-c (дата обращения: 29.07.2022).

13. Смирнова Ю.В. МЕТОДЫ ПРОВЕРКИ УНИКАЛЬНОСТИ ПРОГРАММНОГО КОДА // Вестник магистратуры. 2019. №4-2 (91). https://cyberleninka.ru/article/n/metody-proverki-unikalnosti-programmnogo-koda

14. Статья о работе антиплагиата в Яндекс Контест [Электронный ресурс]. https://yandex.ru/ /support/lyceum-teachers/common-concepts/check-out-solutions.html#plagiarism.

Определение эмоционального состояния обучаемого-ребенка в цифровой образовательной среде по статическим изображениям с применением нейросети

М.С. Дьяченко[1], А.Г. Леонов[2], М.В. Райко[3], А.А. Холькина[4]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, Mdyachenko@niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, rayko@niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kholkina@niisi.ru

Аннотация. В статье описаны исследования возможности распознавания детских эмоций на основе сверточных нейронных сетей для использования в цифровых образовательных средах таких как, например, ЦОС “ПиктоМир”. Рассмотрены альтернативные решения и изложен опыт создания собственной нейронной сети, обученной на экспериментальном наборе данных. Приводятся результаты тестирования системы с помощью различных метрик на основе валидационного набора данных и сравнение полученных результатов с наиболее популярными алгоритмами распознавания эмоций. Сделаны выводы о направлении дальнейших исследований, в частности, необходимость расширения обучающего набора данных, предложены дальнейшие шаги к доработке с целью интеграции в образовательную среду “ПиктоМир”.

Ключевые слова: нейросети, глубокое обучение, распознавание эмоций, классификация эмоций, ПиктоМир.

1. Введение

В последнее время наблюдается интерес к компьютерным программам и системам, учитывающим эмоциональное состояние человека. Сфера применения компьютерного зрения для распознавания эмоций расширяется, и задача становится все более актуальной. На сегодняшний день системы распознавания находят применение в маркетинге, в системах безопасности, в мониторинге состояния водителей, в образовании, в банковской сфере и других областях [1, 2]. В рамках выполненного исследования предпринята попытка сконструировать систему распознавания эмоций детей на основе нейронной сети для использования в цифровой образовательной среде «ПиктоМир», функционирующей на массовых современных планшетах. Мы считаем, что автоматизированные цифровые наблюдения за мимическими эмоциональными реакциями учащегося позволят уловить его настроение, понять, когда он доволен, своими успехами (успешно решил задачу), расстроен из-за ошибок и т.п. Мы надеемся, что полученные данные об эмоциональном состоянии ребенка можно будет использовать для улучшения качества и эффективности обучения.

Авторы ставили целью дифференцировать различные эмоциональные состояния детей, чтобы педагог смог вовремя определить, что ребенок устал, испытывает трудности при выполнении задания. Более того, на изменение настроения ученика может реагировать не только педагог (осведомленный средой ПиктоМир о необходимости вмещательства), но и сама среда ПиктоМир, рекомендую ребенку либо **выполнить** определенные действия вне работы в среде ПиктоМир (взять перерыв на отдых; попросить помощи у преподавателя и т.д.), либо выполнить автоматизированные действия в самой цифровой образовательной среде. Например, посмотреть мультик с подсказкой, пообщаться с виртуальным ассистентом и пр.

2. Обзор значимых публикаций и поиск известных решений

Для решения описанной выше задачи распознавания эмоций, часто используют нейросети – аналогичные тем, что используются для оперирования с аудио, видео и статическими изображениями. Согласно современным исследованиям высокую эффективность в подобных задачах имеют сверточные нейронные сети. Так существуют разработки, способные определять эмоциональное состояние человека по голосу. Например, одно из исследований, посвященное распознаванию эмоций по голосу [3], показало

точность сверточной нейросети 73%. Галичий, Афанасьев и Нестеров в своей работе [4] рассмотрели применение сверточных нейронных сетей для распознавания по статическим изображениям - фото. Нужно отметить, что исследования показали высокие результаты у простейших сверточных сетей, сравнимый со сложными дообученными моделями. К основным выявленным проблемам относятся дисбаланс классов, который необходимо учесть при создании собственного датасета, и переобучение. В аналогичном исследовании по распознаванию эмоций по фотографиям, проведенном на основе сверточной нейросети, точность составила 67,7% [5]. В этой статье указываются также такие проблемы, как отсутствие накопленной объемной базы данных изображений для обучения и трудности при переходе к распознаванию видео (не статичных) изображений.

В классе уже существующих интегрированных разработок был рассмотрен онлайн-сервис распознавания по видео “Emojify” [6], разработанный компанией “Dovetail Labs”. Нейросетевой алгоритм, заложенный в основу системы, позволяет распознавать в реальном времени 6 эмоций с точностью до 99%, используя камеру с телефона или компьютера. Отсутствие полноценной версии в свободном доступе ограничивает использование указанного решения. Аналогичным инструментарием является система, разработанная при поддержке МГУ им. Ломоносова, которая носит название “EmoDetect” [7]. Эта система позволяет распознавать те же 6 эмоций. Однако эта система распространяется в свободном доступе в виде бинарных исполняемых файлов только для ОС Windows, что является потенциальной проблемой для внедрения в “ПиктоМир” и в подобные цифровые образовательные среды.

В процессе поиска уже обученных нейросетей распознавания эмоций был найден набор инструментов “EmoPy” [8] с двумя нейросетями, которые классифицируют эмоции на основе изображений лиц людей. Обучение проводилось на датасете FER+ [9]. Точность нейросети “ConvolutionalLstmNN” на валидации составила 47%, у “TransferLearningNN” всего 29%.

Стоит отметить, что все рассмотренные системы относятся к распознаванию эмоций у взрослых. Эксперименты с распознаванием детских эмоций не описывались, поэтому на данном этапе нет возможности судить о применимости подобных решений относительно детских эмоций без потери качества распознавания.

3. Постановка задач исследования

Поскольку поиски готовых известных решений не привели к желаемому результату в виде готовой модели, которая бы распознавала детские эмоции с хорошей точностью и могла бы быть встроена в цифровую образовательную среду “ПиктоМир”, стала актуальной отдельная работа по разработке подобной модели. В ходе исследования было поставлено три задачи:

1. сформировать набор данных;
2. создать и обучить сверточную нейронную сеть;
3. оценить качество распознавания.

3.1. Набор данных

Для обучения нейросети был собран экспериментальный датасет с фотографиями одного ребенка, который регулярно занимается в образовательной среде “ПиктоМир”. Было получено письменное согласие родителей на проведение исследования и согласие на публикацию в научных статьях фотографий ребенка, сделанных в этом исследовании. Ребенок фотографировал себя на фронтальную камеру и располагал планшет так же, как на занятиях, выражая восемь видов эмоций. Съемка проходила в школе при искусственном и естественном освещении. Экспериментальный датасет включил в себя около 1600 фотографий и 200-230 фотографий на каждый из 8 классов эмоций. Список используемых классов эмоций включал:

1. удивление (surprise);
2. отвращение (disgust);
3. грусть (sad);
4. злость (angry);
5. страх (fear);
6. счастье (happy);
7. презрение (contempt);
8. нейтральное состояние (neutral).

Датасет был поделен на две части: обучающий набор (75%) и валидационный набор (25%). На первом подбирались веса модели, на втором были посчитаны метрики качества.

3.2. Выбор архитектуры нейросети и процесс обучения

Для реализации был выбран язык программирования Python и фреймворк машинного обучения Pytorch. Для решения задачи распознавания эмоций была выбрана предобученная модель на архитектуре сверточной нейронной сети ResNet152. Согласно исследованию [4], метод дообучения заранее обученных моделей сверточных нейросетей с большим количеством слоев (Transfer Learning) весьма эффективен. Модель была предварительно обучена на наборе

данных ImageNet. Он включает в себя более миллиона изображений и 1000 классов, одним из которых является класс «Человек». Она уже содержит веса и смещения, которые представляют особенности того набора данных, на котором она была обучена. Такая модель будет быстрее обучаться на новых данных.

Архитектура сети содержит 152 слоя (см. Рис. 1.). Входной слой, на который подается изображение, имеет размер 224 x 224 x 3, где первые два числа – это ширина и высота изображения, а третье – количество каналов. Далее идут слои со свертками, батч-нормализацией, функцией активации ReLU и макс-пуллингом.

слой	выход	152 слоя
свёртка1	112×112	7×7, 64, шаг 2
свёртка1_x	56×56	3×3 макс-пул, шаг 2 [1×1, 64 3×3, 64 1×1, 256] ×3
свёртка2_x	28×28	[1×1, 128 3×3, 128 1×1, 512] ×8
свёртка3_x	14×14	[1×1, 256 3×3, 256 1×1, 1024] ×36
свёртка4_x	7×7	[1×1, 512 3×3, 512 1×1, 2048] ×3
	1×1	средний пул, 1000-ПС

Рис. 1. Архитектура нейросети ResNet152

Выходной полносвязный слой в предобученной модели имеет 1000 классов, это значение было заменено на 8 классов нового датасета. Чтобы определить принадлежность изображения к классу, значения выходного слоя проходят через функцию аргмакс. Соответственно, результирующим классом является тот, который имеет наибольшее значение на выходе.

Для расширения датасета был применен метод аугментации. Этот метод позволяет добавить на изображения различные искажения, изменения контраста, яркости и сатурации, добавить шумы, повернуть изображения, случайным образом удалить маленькие куски фото. В данной нейросети используется 18 функций аугментации.

Процесс тренировки модели длился 15 эпох с батчами по 32 изображения. В качестве оптимизатора был выбран оптимизатор Адам с гиперпараметром learning rate = 0.0005 (остальные по умолчанию) [10], в качестве функции потерь взята кросс-энтропия. Значение функции потерь на обучающем наборе равнялось 0.013 (см. рис. 2), на валидационном наборе – 0.045 (см. рис. 3). Ниже представлены графики поведения функции ошибки при обучении и валидации.

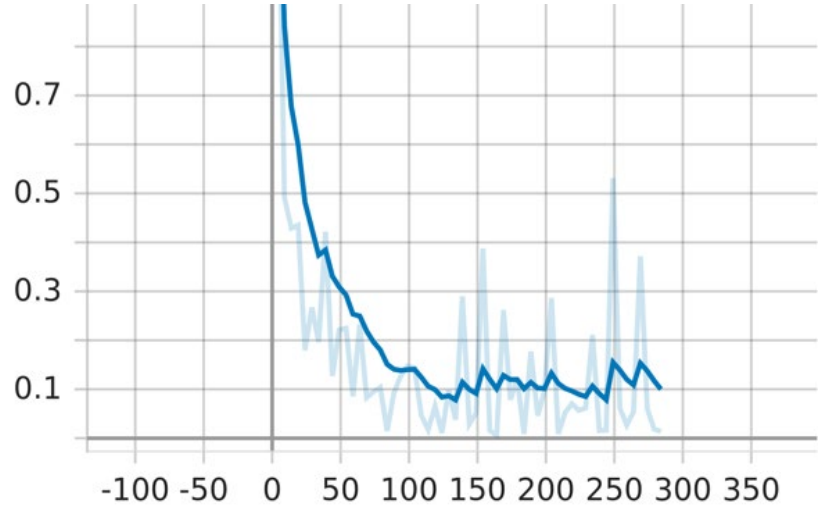

Рис. 2. График ошибки на обучающем наборе

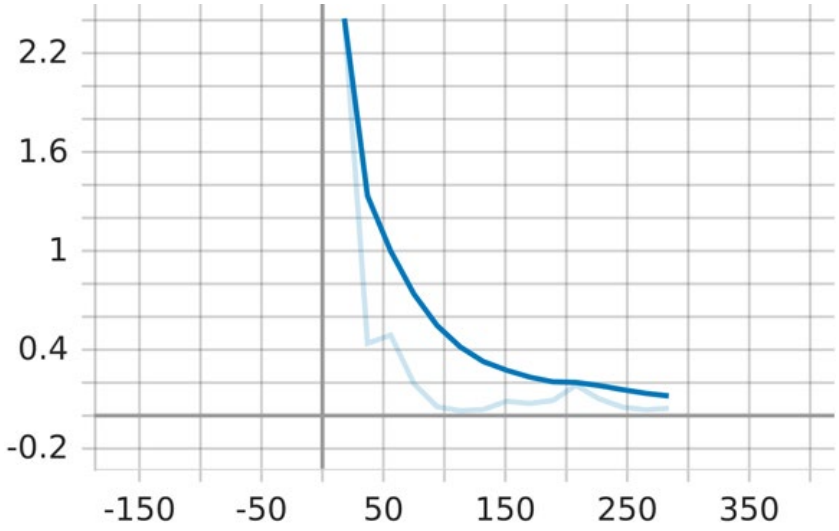

Рис. 3. График ошибки на валидационном наборе

По графикам видно, что у данной нейросети есть потенциал роста, и есть смысл продолжить тренировку для достижения более точных результатов.

3.3. Оценка качества распознавания

Для оценки качества распознавания использовались confusion matrix (матрица ошибок), precision (точность) и recall (полнота).

Для обученной модели распознавания эмоций была получена следующая матрица ошибок:

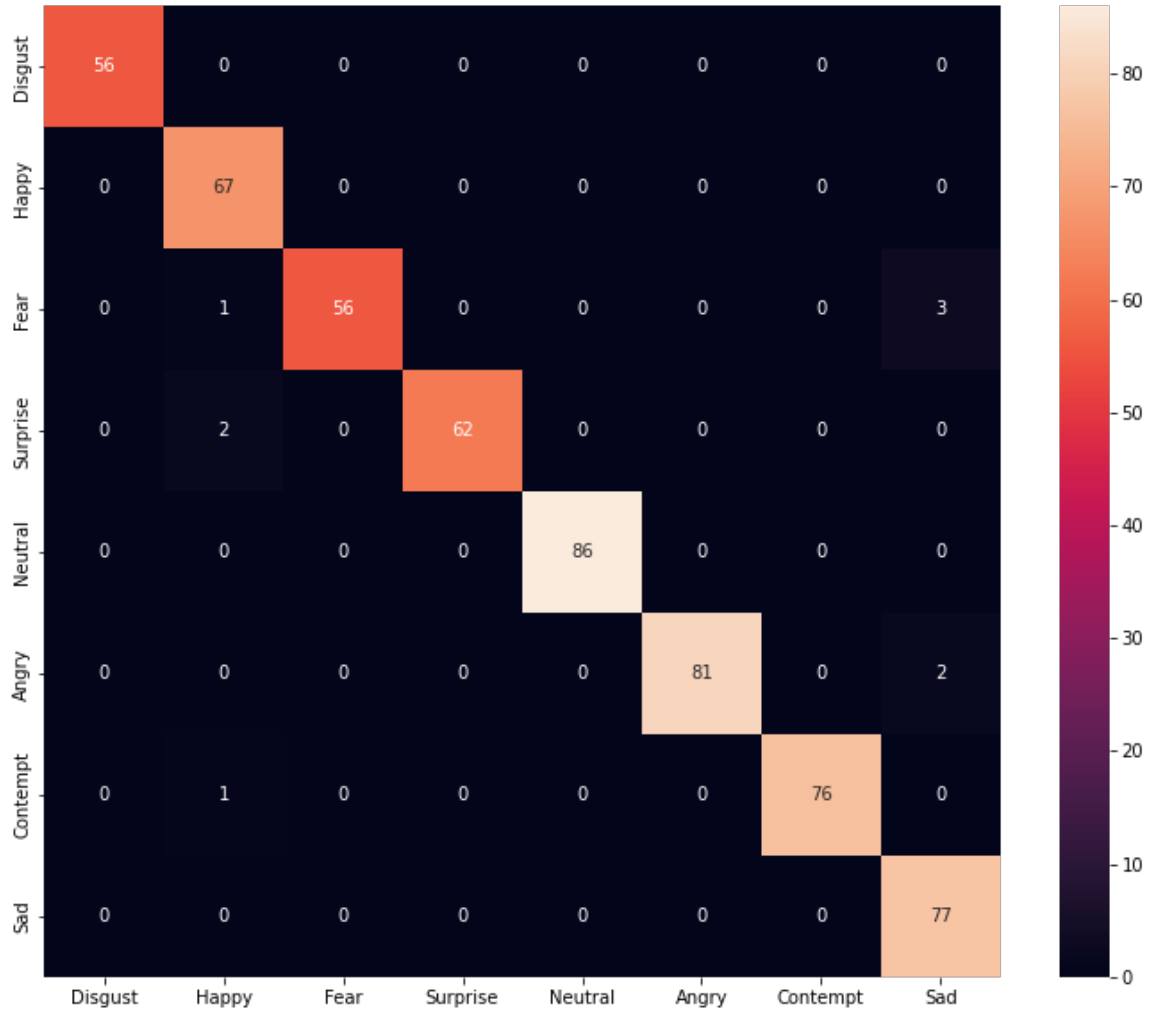

Рис. 4. Матрица ошибок

Напомним, что по горизонтали — прогноз модели, по вертикали — истинные классы. Можно заметить, что модель предсказывает довольно точно. Чаще всего модель ошибалась в предсказании класса “Fear” - 4 раза, чуть меньше в классах “Surprise” и “Angry” - 2 раза, и один раз в классе “Contempt”. Ниже приведены примеры верной классификации.

Рис. 5. Предсказанный класс – “Happy”

Рис. 6. Предсказанный класс – “Neutral”

Рис. 7. Предсказанный класс – “Sad”

Для вычисления precision и recall были взяты средние значения по всем классам.

Mean precision = 0.985
Mean recall = 0.983

Значения говорят о высокой точности модели, но стоит отметить, что обучение проходило на небольшом объеме данных, поэтому при расширении обучающего датасета ожидается снижение точности. По итогам обучения можно сделать вывод, что архитектура нейросети ResNet152 подходит под задачу и её можно использовать в дальнейших исследованиях.

3. Результаты и выводы

В проведенном исследовании была проведена опытная практическая работа по созданию системы распознавания эмоций на основе сверточной нейронной сети. Выбранная модель зарекомендовала себя достаточно быстрым и эффективным дообучением. На валидационной выборке точность составила 98,5%, однако, стоит заметить, что обучение и проверка результатов проводились на относительно небольшом датасете. В дальнейшем планируется расширение датасета, которое будет учитывать такие нюансы, как различное освещение, различные движения головы и различия в чертах лица из-за этнической принадлежности, возраста, пола, и очков, а также обучение нейросетей на других архитектурах для достижения еще большей точности распознавания.

Из-за ограниченности бюджета данного исследования, использовался датасет фотографий одного единственного ребенка. Безусловно, следует проверить результат работы построенной нейронной сети распознавания эмоций на достаточно представительном множестве детей возраста 6-8 лет. При достижении качественного распознавания (с точностью 70-80%), нейросеть планируется встроить в цифровую образовательную среду “ПиктоМир”.

Авторы благодарны администрации школы 199 г. Москвы и ученице 3 класса этой школы Арине Пушиной за помощь в организации и проведении исследования, описанного в настоящей статье.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0010.

Determining the Emotional State of a Student-Child in a Digital Educational Environment From Static Images Using a Neural Network

M.S. Diachenko, A.G. Leonov, M.V. Rayko, A.A. Kholkina

Abstract. The article describes research on the possibility of recognizing children's emotions based on convolutional neural networks for use in digital educational environments such as, PiktoMir. Alternative solutions are considered and the experience of creating your own neural network trained on an experimental data set is presented. The results of testing the system using various metrics based on the validation data set and comparing the results with the most popular emotion recognition algorithms are presented. Conclusions are drawn on the direction of further research, in particular, the need to expand the training data set, further steps are proposed for refinement in order to integrate into the PiktoMir educational environment.

Keywords: neural networks, deep learning, emotion recognition, classification of emotions, PiktoMir.

Литература

1. Е. Хрисанфова. Независимое издание о технологиях и бизнесе RB.RU [Электронный ресурс]: Эмоциональный ИИ: кто и зачем распознаёт эмоции в России и за рубежом. https://rb.ru/longread/emotion-ai/.

2. Ю. Н. Харари. 21 урок для XXI века. Синдбад, 2019. с. 52.

3. А Сергунов, Д.И., А.А. Артёмова и С.С. Гришунов. Система распознавания эмоций по голосу на основе сверточной нейронной сети. E-Scio, 7, 2019. https://cyberleninka.ru/article/n/sistema-raspoznavaniya-emotsiy-po-golosu-na-osnove-svertochnoy-neyronnoy-seti/viewer.

4. Галичий, Д.А., Г.И. Афанасьев и Ю.Г. Нестеров. Распознавание эмоций человека при помощи современных методов глубокого обучения. E-Scio, 5, 2019. https://cyberleninka.ru/article/n/raspoznavanie-emotsiy-cheloveka-pri-pomoschi-sovremennyh-metodov-glubokogo-obucheniya/viewer.

5. M. Singh, S. K. Sharma, S. Paul, J. P Sajeevan, S.Paul. Facial emotion recognition system. IJARIIE-ISSN(O)-2395-4396, Vol-6 Issue-6, 2020. https://www.researchgate.net/publication/347797356_FACIAL_EMOTION_RECOGNITION_SYSTEM

6. Сайт сервиса “Emojify” [Электронный ресурс]. https://emojify.info/menu. (дата обращения: 01.08.2022)

7. Сайт сервиса “EmoDetect” [Электронный ресурс]. https://emodetect.ru. (дата обращения: 01.08.2022)

8. Репозиторий “EmoPy” [Электронный ресурс]. https://github.com/thoughtworksarts/EmoPy. (дата обращения: 01.08.2022)

9. Набор данных с эмоциями “FER+” [Электронный ресурс]. https://github.com/Microsoft/FERPlus. (дата обращения: 01.08.2022)

10. Оптимизатор Adam [Электронный ресурс]. https://pytorch.org/docs/stable/generated/torch.optim.Adam.html. (дата обращения: 01.08.2022)

11. Иванов, И.А., Е.А. Сопов, И.А. Панфилов. Многокритериальный подход к проектированию ансамбля нейросетевых классификаторов с отбором информативных признаков для решения задачи распознавания эмоций. Сибирский аэрокосмический журнал, 4, 2015. https://cyberleninka.ru/article/n/mnogokriterialnyy-podhod-k-proektirovaniyu-ansamblya-neyrosetevyh-klassifikatorov-s-otborom-informativnyh-priznakov-dlya-resheniya/viewer.

12. Курицкий, В.Ю., С.В. Садов. Нейросетевой алгоритм распознавания эмоций человека по изображению лица. Компьютерные технологии и анализ данных (CTDA’2020): материалы II Междунар. науч.-практ. конф., Минск, 2020: БГУ https://elib.bsu.by/bitstream/123456789/248683/1/245-248.pdf.

О подходах к построению надежной цифровой образовательной платформы

М.С. Дьяченко[1], М.А. Кузьменко[2], А.Г. Кушниренко[3], Г.О. Райко[4], И.Г. Райко[5]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, Mdyachenko@niisi.ru;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, gmk@niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, agk_@mail.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, raiko@niisi.msk.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, ilya.rayko@niisi.ru

Аннотация. В нашей стране активно идет процесс цифровой трансформации образования, который выдвигает требования создания цифровых образовательных платформ, обеспечивающих надежный доступ к разнообразным цифровым образовательным ресурсам. Такие платформы по сути своей являются распределенными вычислительными системами. Важность и критичность задач, выполняемых цифровыми образовательными ресурсами, определяет высокий уровень технических требований к платформам, на которых они реализованы. В статье рассмотрены критерии, обеспечивающие надежность платформ, проанализированы существующие популярные решения в области обеспечения надежности цифровых образовательных платформ и проведен анализ соответствия этих решений указанным критериям. Показано, что существующие цифровые образовательные платформы, в основном далеки от соответствия поставленным требованиям. Авторами предложен вариант цифровой образовательной платформы, удовлетворяющей сформулированным критериям.

Ключевые слова: цифровая образовательная платформа, вычислительные системы высокой надежности, территориально-распределенные вычислительные системы, цифровая образовательная платформа Мирера

1. Введение

Россия является одной из лидирующих стран в области цифровизации финансов, экономики, образования, госуслуг, а также активно проводит цифровую трансформацию образования во всех его составных частях, от цифрового хозяйства университетов до цифровизации образовательного процесса на уровне территориальных и федеральных органов власти. Тот факт, что Россия является самым большим по территории государством в мире, накладывает определенные требования к подобным цифровым платформам, в том числе является необходимым обеспечение высокоскоростного и надежного доступа к ресурсам вне зависимости от места нахождения обучаемого. В век тотальной цифровизации потребности внедрения цифровых образовательных платформ растут, однако основы педагогических образовательных систем были заложены еще в прошлом веке [1].

Для соответствия растущим потребностям в области цифровизации образования были разработаны десятки систем, многие из которых не отвечают современным требованиям. Особенно выпукло это несоответствие проявилось во время пандемии Covid-19, которая привела к крупнейшему за всю историю сбою в функционировании систем образования [2].

Во многом напряженность ситуации с надежностью цифровых образовательных платформ связана с неправильной оценкой требований к отказоустойчивости и высокой готовности таких систем и смешением этих требований с требованиями по распределенности нагрузки. Особенно ярко эта проблема видна среди систем проведения олимпиад и соревнований по программированию разных уровней, которые долгое время считались передовыми среди различных образовательных платформ.

2. Основные принципы

Цифровая образовательная платформа является распределенной системой программного обеспечения с отслеживаемым состоянием. Такие системы должны удовлетворять требованиям *готовности* и *согласованности*.

Вычислительная система обладает *высокой готовностью*, если она может продолжать работать при отказе некоторых компонент. Системы разделяют по количеству отказов, которые они способны выдержать. Ввиду своей специфики образовательные платформы обязаны сохранять высокую готовность всегда, даже во время проведения обслуживающих работ. Поэтому такие системы должны проектироваться с возможностью работы при двух и более отказах. В таких

системах должны использоваться отказоустойчивые кластеры класса не хуже HA-2 (HA это сокращение от High-Availability cluster - Отказоустойчивый кластер).

Основой обеспечения высокой готовности любой системы является условие, что *среднее время восстановления системы* намного меньше *среднего времени между отказами* системы: при нарушении этого условия текущее количество отказов в системе будет расти.

Говорят, что распределенная система обладает *согласованным состоянием*, если в каждый момент данные о состоянии некоторого хранящегося в системе логического объекта, полученные от разных компонент системы, совпадают. В некоторых случаях это требование ослабляют, допуская вре́менное расхождение в данных, полученных от разных компонент. К сожалению, такое ослабление не является допустимым для цифровых образовательных платформ.

Связи между готовностью и согласованностью системы описываются эвристическим утверждением, которое специалисты в теории распределенных систем несколько вольно называют CAP-теоремой [3]. Сочетание латинских букв CAP произведено от трех английских терминов, относящихся к распределенным системам: согласованность данных (Consistency), доступность данных (Availability) и устойчивость к разделению сети (Partitioning). Согласно CAP-теореме, любая распределенная система в случае ее разбиения на части в результате какого-либо отказа может либо продолжать работать (быть доступной пользователям), либо поддерживать согласованность данных.

Для достижения надежности системы должны существовать стратегии по восстановлению после катастроф. Такие стратегии связывают со следующими двумя величинами: временем на восстановление работоспособности (RTO) и интервалом времени, данные за который могут быть утеряны (RPO) (Рис. 1). У надежной цифровой платформы эти показатели должны быть близки к нулю.

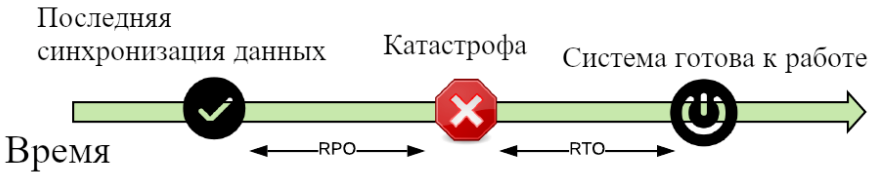

Рис. 1. RTO и RPO

Все эти требования выполняются для активно-активных территориально-распределенных систем. Такие системы являются набором удаленных комплексов (кластеров), каждый из которых является распределенной вычислительной системой. В отличие от активно-пассивных систем, в которых все кластеры, кроме одного, во время нормальной работы находятся в резерве, в активно-активной схеме все кластеры обслуживают запросы клиентов. Из-за этого удается достичь околонулевого значения RTO: клиент быстро находит другой активный кластер и продолжает нормальную работу. Для предотвращения разбиения системы в случае катастроф (и потери готовности и согласованности, согласно CAP-теореме) *такие системы должны включать в себя хотя бы три кластера*. В таком случае низкий RPO достигается посредством строгой согласованности данных между кластерами в системе (что требует наличия высокоскоростного канала связи). Для поддержания условия HA-2 *каждый кластер должен состоять из хотя бы пяти вычислительных узлов*.

3. Имеющиеся решения

Ниже представлен обзор имеющихся решений.

3.1 Система *Programming Contest Control, PC²*

Система PC² [4] — разработка Университета штата Калифорния, США (California State University, CSUS) в Сакраменто. Система использовалась при проведении Международной студенческой олимпиады по программированию (International Collegiate Programming Contest), проводимой под эгидой Ассоциации вычислительной техники (Association for Computing Machinery, ACM inc.) и ряда других мероприятий. Система имеет модульную структуру. За хранение всех конфигурационных данных соревнований, прием решений от участников соревнований и хранение таких решений и ведение коммуникационных соединений между всеми остальными модулями системы отвечает модуль Server.

Система PC² допускает собственное географическое распределение, в каждом месте прове-

дения соревнований должен быть запущен экземпляр модуля типа Server. Однако, такая конфигурация решает лишь задачу минимизации задержек сети при обмене данными между географически распределенными точками. В системе должен существовать один экземпляр модуля типа Server, который должен быть объявлен основным, остальные модули типа Server считаются вспомогательными. Другие модули системы также допускают функционирование на выделенных аппаратных ресурсах, но лишь в целях распределения нагрузки на аппаратуру, минимизации нагрузки на единый сервер. Таким образом, средства распределения вычислений, предоставляемые системой PC^2, решают задачу проведения соревнований в реальном масштабе времени.

В целом, система не решает задач обеспечения отказоустойчивости или высокой готовности.

3.2 Система Contest Management System (CMS)

Система Contest Management System [5], CMS создана группой независимых разработчиков. Данная система использовалась при проведении Международной олимпиады по информатике (International Olympiad in Informatics, IOI) и ряда других мероприятий.

Система имеет модульную структуру, сервисы системы могут быть запущены на разных серверах, допускается репликация сервисов на различных серверах. Система использует СУБД PostgreSQL для хранения различных состояний соревнования, использование иных СУБД невозможно, поскольку CMS существенно опирается на специальный тип PostgreSQL «большой двоичный объект» (Binary Large Object, BLOB). Функции обеспечения отказоустойчивости, обеспечиваемые CMS также существенно опираются на соответствующие возможности СУБД PostgreSQL. К преимуществам такого подхода следует отнести то, что до тех пор, пока база данных находится в корректном состоянии, сервисы CMS могут функционировать независимо, в частности, в любой момент времени любой сервис может быть остановлен и перезапущен на другом сервере. При этом отсутствует необходимость в явном переносе информации между серверами. Сама СУБД PostgreSQL должна быть сконфигурирована так, чтобы предоставлять функции резервирования и отказоустойчивости. Следует отметить, что для PostgreSQL существуют разнообразные широкораспространенные решения для обеспечения резервирования и отказоустойчивости.

Все основные сервисы системы CMS спроектированы так, что любой из сервисов может быть остановлен и перезапущен без потери целостности данных и без блокирования других сервисов системы. Система CMS обладает всеми требуемыми средствами обеспечения отказоустойчивости. Потенциальные проблемы, возникающие из-за существования в системе единой точки отказа, СУБД PostgreSQL, могут быть эффективно решены на уровне самой СУБД.

3.3 Система ejudge

Система ejudge [6] разрабатывается на факультете вычислительной математики и кибернетики Московского государственного университета имени М. В. Ломоносова. Система используется при проведении различных олимпиад по программированию в России, в частности, при проведении регионального этапа Всероссийской олимпиады школьников по информатике и программированию в некоторых регионах.

В качестве системы управления базой данных ejudge использует СУБД MySQL или MariaDB. Интерфейсные компоненты функционируют в рамках HTTP-сервера Apache и запускаются при поступлении запроса HTTP протокола, инициируемого клиентами.

Несмотря на то, что общесистемное программное обеспечение промежуточного уровня, включая СУБД и HTTP-сервер имеет развитые средства обеспечения отказоустойчивости, архитектура системы ejudge не позволяет полноценно задействовать данные функции. Управляющие компоненты работают в одном экземпляре, для обмена данными между некоторыми управляющими компонентами используются средства локальных коммуникаций — сокеты типа UNIX. Следует отметить, что второй тип взаимодействия между управляющими компонентами — структура общих (разделяемых) каталогов, хотя и допускает использование распределенных файловых систем, функционирующих по протоколам NFS или SMB/CIFS, настройка функций отказоустойчивости для подобных файловых систем требует штата системных администраторов очень высокой квалификации.

В целом, следует констатировать, что система ejudge не обладает функциями отказоустойчивости или высокой готовности, несмотря на наличие таких средств у отдельных частей системы.

3.4 ЦОП Мирера

Цифровая образовательная платформа Мирера [7], разрабатываемая специалистами ФГУ ФНЦ НИИСИ РАН, состоит из нескольких компонент:

– Система аутентификации пользователей;

– Система проверки решений учеников;

– API сервер для выдачи и приема решений задач;

– HTTP-сервер для создания, редактирования индивидуальных заданий, отслеживания и хранения результатов выполнения заданий.

Платформа является автономной и легко масштабируемой, как вертикально, так и горизонтально. Это обусловлено выбором технологий и программных продуктов, поддерживающих маштабируемость и отказоустойчивость. Отслеживаемое состояние имеется только у двух компонент: базы данных и очереди проверки (Рис. 2).

В качестве базы данных используется нереляционная документоориентированная СУБД MongoDB [8]. Все файлы, необходимые для работы системы, хранятся в модуле Grid File System – модулем MongoDB для работы с файлами. Данный модуль позволяет эффективно работать с файлами, а также позволяет использовать стандартные механизмы базы данных для обеспечения отказоустойчивости и масштабирования.

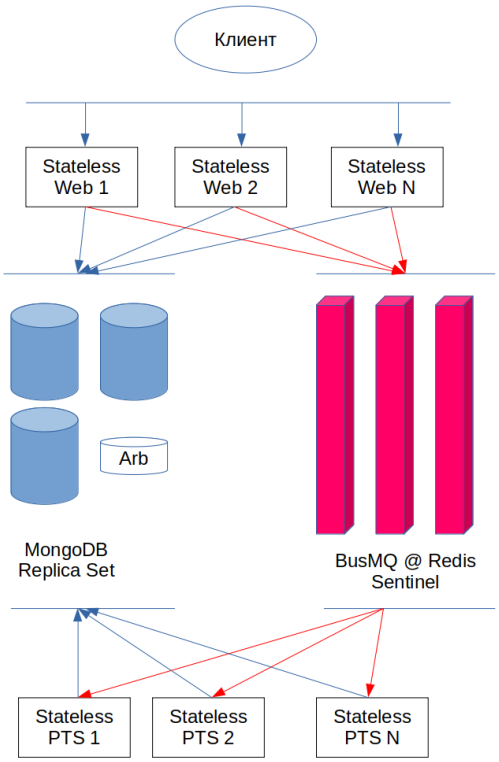

Рис. 2 Схема ЦОП Мирера. PTS — система проверки решений, в Web-сервисах находятся все системы, взаимодействующие с клиентами.

СУБД MongoDB является распределенной системой управления базой данных, поддерживающей механизм репликации данных — критического механизма для обеспечения функций отказоустойчивости. В терминах MongoDB реплика представляет собой группу нескольких экземпляров главного серверного процесса СУБД (mongod), каждый из которых хранит копию данных. При этом один из экземпляров объявлен главным, он ответственен за подтверждение операций записи данных и ведение журнала изменений набора данных. Остальные экземпляры являются вспомогательными, их задача — поддерживать собственные реплики данных в актуальном состоянии. В случае сбоя главного экземпляра группы вспомогательные экземпляры проводят выборы нового главного экземпляра. Дополнительно в группе может быть назначен арбитр — специальный экземпляр главного серверного процесса СУБД, на который не возложена задача хранения данных. Задача арбитра — обеспечение кворума для голосования при выборе нового главного экземпляра.

Помимо механизма репликации данных в СУБД MongoDB используется метод горизонтального масштабирования посредством техники сегментирования данных (sharding). В технике сегментирования хранимые данные разбиваются на диапазоны (сегменты) с помощью

ключа сегментирования. Каждый сегмент обслуживается своим экземпляром сервера СУБД. Техника горизонтального масштабирования обеспечивает дополнительный уровень доступности данных: при отказе одного из серверов СУБД теряется лишь порция данных, соответствующая одному сегменту.

Модуль Grid File System (GridFS) реализует файловую систему на базе СУБД MongoDB. В парадигме GridFS файл хранится не единой сущностью, но представляет набор частей (chunks), хранимых в виде отдельных документов (в терминологии MongoDB), т. е. иерархически организованных структур данных. При этом каждый файл представляется двумя коллекциями документов: одна коллекция хранит собственно части файла, другая — метаданные.

Очередь проверки построена на базе библиотеки BusMQ (модуля Node.js), в свою очередь основанной на сетевом журналируемом хранилище данных Redis [9]. Очередь BusMQ обеспечивает гарантированную доставку сообщений и поддерживает кластеризованный вариант сервера Redis - Redis Sentinel. Помимо этого, Redis Sentinel в случае отказа любой из систем сохраняет информацию об очереди, что гарантирует отсутствие потерь сообщений о тестировании.

Redis Sentinel предоставляет функции отказоустойчивости и высокой готовности для хранилища данных Redis. Следует отметить, что хранилище данных Redis само по себе содержит базовые элементы для построения отказоустойчивых решений. В частности, помимо механизмов поддержки транзакционных и пакетных операций над хранилищем, ведения снимков данных и журналирования для обеспечения постоянного хранения данных, Redis поддерживает репликацию типа master-slave, когда в системе существует выделенный основной сервер (master), обрабатывающий текущие запросы, и один или несколько вспомогательных серверов (slave), которые содержат копию текущих данных. Технология Redis Sentinel имеет распределенную архитектуру без единой точки отказа. В системе может существовать несколько серверов Redis Sentinel, взаимодействующих между собой для обеспечения, как функции постоянной готовности, так и механизмов голосования при принятии решения об отказе основного сервера Redis, когда несколько экземпляров Redis Sentinel должны прийти к выводу об отказе.

4. Выводы

Проведенные исследования показывает, что существующие решения задач цифровизации образования при построении цифровых образовательных платформ не удовлетворяют требованиям надежности. Возникающие из-за этого проблемы могут иметь далеко идущие последствия вплоть до масштабных срывов образовательного процесса на уровне субъекта РФ или даже на федеральном уровне. Сформулированные авторами критерии могут стать основными критериями как при разработке георасределенной системы Мирера, так и при построении других цифровых образовательных платформ.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0010.

On Approaches for Building Reliable Digital Educational Platforms

M.S. Diachenko, M.A. Kuzmenko, A.G. Kushnirenko, G.O. Raiko, I.G. Raiko

Abstract. In our country, the process of digital transformation of education is actively underway, which puts forward the requirements for the creation of digital educational platforms that provide reliable access to a variety of digital educational resources. Such platforms are essentially distributed computing systems. The importance and criticality of the tasks performed by digital educational resources determine the high level of technical requirements for the platforms on which they are implemented. The article considers the criteria that ensure the reliability of platforms, analyzes the existing popular solutions in the field of ensuring the reliability of digital educational platforms, and analyzes the compliance of these solutions with the specified criteria. It is shown that the existing digital educational platforms are generally far from meeting the set requirements. The authors proposed a variant of a digital educational platform that meets the formulated criteria.

Keywords: digital educational platform, high-available cluster, geographically distributed cluster, digital educational platform Mirera

Литература

1. А.Г.Леонов, Ю.А.Первин. Качественные оценки эффективности методики обучения элементам информатики в пропедевтическом курсе. «Ярославский педагогический вестник», (2015), № 5.

2. Концептуальная записка: Образование в эпоху COVID-19 и в последующий период, Организация Объединенных Наций, август 2020 г. [Электронный ресурс] https://www.un.org/sites/un2.un.org/files/policy_brief_-_education_during_covid-19_and_beyond_russian.pdf (дата обращения: 29.08.2022).

3. Seth Gilbert, Nancy Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, ACM SIGACT News, Volume 33, Issue 2, June 2002

4. Steve Baber, David Hoelzeman, Becky Cunningham, Rick Massengale, Programming contest hosting: conference tutorial, Journal of Computing Sciences in Colleges, Volume 26, Issue 5, May 2011

5. Stefano Maggiolo, Giovanni Mascellani, Introducing CMS: A Contest Management System, Olympiads in Informatics, Vol.6, 2012

6. Ejudge – система для проведения различных мероприятий, в которых необходима автоматическая проверка программ. [Электронный ресурс] https://ejudge.ru/wiki/index.php/Система_ejudge (дата обращения: 29.08.2022).

7. Н.О.Бесшапошников, А.Г.Кушниренко, А.Г.Леонов, К.А.Прокин, Построение отказоустойчивых систем для проведения олимпиад по программированию, Труды НИИСИ РАН, Том 8, №6, 2018

8. MongoDB Operations Best Practices. [Электронный ресурс] https://s3-ap-southeast-1.amazonaws.com/tv-prod/documents%2Fnull-10gen-MongoDB_Operations_Best_Practices.pdf (дата обращения: 29.08.2022).

9. High availability with Redis Sentinel. [Электронный ресурс] https://redis.io/docs/management/ /sentinel/ (дата обращения: 29.08.2022).

Подходы к учету посещаемости студентов в цифровой образовательной платформе Мирера

А.Г. Леонов[1], К.А. Мащенко[2], А.В. Шляхов[3], А.А. Холькина[4]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[2] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kirill.mashchenko@vip.niisi.ru;

[3] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, shlyakhov@vip.niisi.ru;

[4] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kholkina@niisi.ru

Аннотация. При использовании образовательных платформ оперативность и эффективность содержательных взаимодействий преподавателя с его студентами заметно улучшаются, однако при этом остается недостаточно цифровизированным ряд рутинных процессов, которые отнимают время и силы педагога. К числу таких рутинных процессов относится учет посещаемости студентов. Учет посещаемости должен производиться педагогом вне зависимости от того, проводится ли занятие очно или дистанционно. Цифровизированный процесс учета посещаемости должен быть удобен для преподавателя, минимизируя время и усилия, затраченные преподавателем на регистрацию присутствующих студентов на каждом занятии. Процесс должен быть удобен и для самих обучающихся, не создавая дискомфортных ситуаций для студентов. В статье предлагается вариант решения задачи учета посещаемости на примере авторской цифровой образовательной платформы Мирера. Предлагаются сценарии использования механизма учета посещаемости для увеличения вовлеченности студентов в процесс обучения.

Ключевые слова: цифровая образовательная платформа, цифровая образовательная среда, Мирера, посещаемость, видеоконференция, геопозиция, нейронные сети

1. Введение

Цифровая трансформация образовательных процессов подразумевает не только создание цифровых образовательных курсов или формирование гипертекстовых учебных материалов [1]. Несомненно, что процесс цифровой трансформации образовательных процессов должен охватывать также и рутинные вспомогательные, предметно независимые процессы, в первую очередь важный процесс учета и контроля посещаемости студентов. При этом желательно использовать автоматизацию рутинных процессов для повышения качества и эффективности обучения. Чтобы соответствовать современным требованиям к качеству образования, преподаватель не должен в процессе обучения отвлекаться и тратить драгоценное время занятия на рутинные действия по учету посещаемости, а студент в свою очередь не должен волноваться о том, что система учета посещаемости зафиксирует некорректную информацию о пропуске студентом какого-либо занятия. При этом нужно отдавать себе отчет в том, что контроль за посещаемостью в первую очередь призван помочь студентам планировать свое время и справляться с учебной нагрузкой без авралов и сбоев. Педагог должен быть уверен, что студент не может сфальсифицировать результаты учета посещаемости, поскольку подобная фальсификация, по какой бы причине она ни была осуществлена студентом, с большой вероятностью приведет к негативным последствиям для самого студента. Организация надежной и удобной системы учета посещаемости – задача любой современной цифровой образовательной платформы.

2. Известные подходы к решению задачи учета посещаемости

В настоящее время существуют различные способы автоматизации учета посещаемости. Одной из них является система контроля посещаемости на основе RFID-меток или электронных пропусков у каждого студента и сотрудника [2]. Возможности данной системы обширны, однако с помощью неё можно контролировать только время входа и выхода в университет, но нельзя действительно отследить, присутствовал ли человек на конкретном занятии, так как для этого потребовалось бы установить RFID-турникеты во все учебные аудитории. Помимо этого, данная система требует выделения большого

бюджета на ее установку и последующее содержание. Кроме того, подобные системы не предоставляют преподавателям удобный оперативный доступ к результатам учета. Таким образом, при больших финансовых вложениях данная система не дает удобного результата, интегрированного в другие образовательные системы, которыми пользуется преподаватель.

Другим способом автоматизации процесса учета посещаемости студентов являются методы, основанные на автоматическом распознавании лиц [3]. При данном подходе анализируется видеопоток со студентами, из него убираются шумы и выделяются лица студентов, после чего они распознаются и определяется, какой студент изображен в данном видеокадре. Однако у этого подхода есть несколько минусов. Система часто работает неудовлетворительно, если в аудитории плохое освещение. Помимо этого, когда много студентов одновременно заходит в аудиторию, не избежать перекрытия лиц одних студентов лицами других. Такой же проблемой является поворот или наклон головы человека. Также требуется просить студентов не выражать эмоции, которые мешают правильному распознаванию, снимать очки, шарфы и другие аксессуары и т.п. Для тренировки модели выделения конкретного студента на фото и видео потоке, потребуется собирать и хранить большое количество фотографий в разных ситуациях, поэтому нужно учитывать закон о сборе и хранении персональных данных. В период пандемии или локальных эпидемий гриппа, ситуация усложняется еще и тем, что студенты могут быть обязаны носить медицинские маски. И даже если все эти условия выполнены – все равно не избежать ошибок в определении наличия в аудитории человека в принципе и ошибочного определения человека на фотографии, при котором будет засчитано посещение студента, которого нет, и не засчитано посещение студента, который на самом деле был. И в случае ошибочного определения быстро обнаружить и решить данную проблему не получится.

Из других решений в настоящий момент на рынке можно выделить несколько компьютерных систем учета посещаемости:

1. Автоматизированный комплекс учета посещаемости и успеваемости студентов «АРАТО»;
2. Модуль автоматизированного комплекса «Альма Матер», под названием «Учет успеваемости и посещаемости»;
3. Автоматизированная система на базе «1С: Предприятие 7.7», под названием «1С: Деканат».

АК «АРАТО» предназначен для использования в учебном процессе учебного заведения. Комплекс обеспечивает ведение электронного журнала, создание отчетов, и ведение база данных информации об учащихся. Студенты заносятся в БД и прикрепляются к определенной группе. Затем, при проведении занятия, преподаватель вручную отмечает присутствовавших студентов, что и является главным минусом данной системы.

АК «Альма Матер» также позволяет учитывать посещаемость и успеваемость. В дополнении к этому он позволяет своевременно уведомлять деканат о пропуске студентами большого количества занятий. Однако посещаемость учитывается не просто проставлением соответствующего бита в записи данного студента, а формируется понедельно на основе документальных данных путем обработки информация о пропущенных занятиях. Для ввода предусмотрено создание по каждой группе студентов множество вкладок, которое соответствуют количеству и номерам семестров из учебного плана. По каждому студенту группы вводится общее за неделю по всем дисциплинам количество пропущенных часов, что предполагает ведение отдельного журнала в течение недели для учета текущей посещаемости.

Автоматизированная система «1С: Деканат» на базе «1С: Предприятие 7.7» покрывает учет посещаемости и хорошо интегрирована с другими системами по типу «1С: Библиотека», «1С: Бухгалтерия», однако у нее нет интеграции с системами, в которых студенты сдают задания, и сама система предназначена больше для деканата, нежели для преподавателей.

Главным недостатком всех описанных систем является то, что для учета посещаемости преподаватель должен напрямую провзаимодействовать с каждым студентом, что не только отнимает время от занятия, но и отвлекает от его проведения. Состоятельность данных систем в формате дистанционного обучения тоже остается под вопросом, и вся организационная часть ложится на плечи преподавателя.

3. Идеи решения, предлагаемого в системе Мирера

К учету посещаемости можно подходить различными способами [4]. На первый взгляд кажется удачной идея добавления к каждому контесту, где требуется учет посещаемости, специального задания «Селфи», в котором студенту нужно отправить свою фотографию, а преподаватель, увидев это, отмечает его в журнале. Однако у данного подхода есть ряд минусов, например, студент может сделать некоторое количество фотографий заранее и в дальнейшем использовать их для прохождения данного вида проверки. Также это не избавляет преподавателя

от «ручной» проверки информации, предоставленной каждым студентом на каждом контесте.

Следующей идеей, которая теоретически могла бы быть реализована, является выдача студентам QR-кода, отсканировав который, студент автоматически отмечался бы в журнале посещаемости. Однако тогда очевидным методом обхода системы учета будет являться передача персонального QR-кода прогульщика одногруппникам, посещающим данное занятие. Можно усовершенствовать систему и выдавать студентам персональные QR-коды, размещенные на их фотографиях, но и это не освобождает преподавателя от взаимодействия с каждым студентом лично. Помимо этого, для реализации данного подхода все студенты должны иметь с собой мобильный телефон или планшет для сканирования кода, что тоже может вызывать некоторые трудности, особенно в контексте того, что все задания выполняются на компьютерах.

4. Учет посещаемости в офлайн формате

Учитывая все вышеизложенные проблемы, было принято решение использовать в ЦОП Мирера [5] следующую схему для учета посещаемости студентов в офлайн формате: преподаватель, приходя в аудиторию, отправляет в контест информацию о своей геопозиции при помощи сервиса Geolocation API [6], предоставляемого всеми современными браузерами, а именно долготу, широту и точность определения местоположения в метрах (рис. 1).

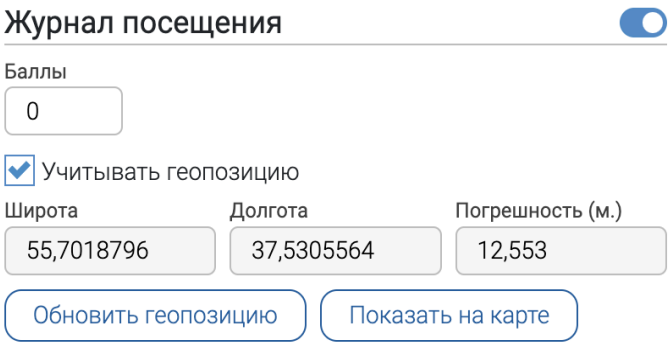

Рис. 1. Отправка геолокации преподавателем

Затем студенты, заходя в аудиторию, аналогично отправляют на сервер свою геолокацию, а в ответ получают информацию, засчитано ли им посещение данного контеста (рис. 2).

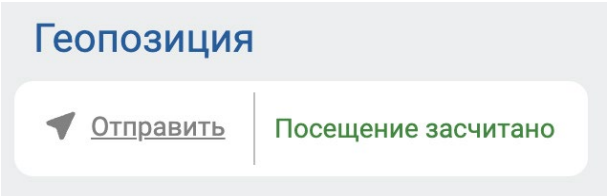

Рис. 2. Отправка геопозиции со стороны студента

При таком подходе теоретически остается возможность фальсификации путем передачи прогульщиком своего смартфона одногруппнику, посещающему занятие, но практическое использование подобного механизма фальсификации маловероятно.

Для подтверждения результатов опроса посещаемости можно использовать простой пересчет людей в аудитории: как ручной, так и автоматизированный. Для этого в помощь преподавателю был разработан дополнительный инструмент – Telegram-бот, который по фотографии аудитории определяет и подсчитывает количество находящихся в ней людей.

С решением задачи детекции человека на фотографиях с каждым годом все лучше справляются нейросети [7]. Нами было протестировано два подхода: детекция людей и детекция лиц. Для этого использовались предобученные сверточные нейронные сети детекции YOLOv5 [8] и MTCNN [9]. Первая нейросеть обучена на датасете COCO и детектирует 80 классов, среди который есть класс “person”. Данная нейросеть демонстрирует высокое качество детекции и за

счет своего однопроходного алгоритма работает быстрее, чем многопроходые аналоги [10]. Для того, чтобы нейросеть определяла только людей, она была дополнительно настроена на определение только этого класса. Вторая нейросеть MTCNN обучена детектировать только лица. Она предобучена на большом датасете и имеет высокое качество детекции (около 99% accuracy). Таким образом, необходимо было проверить, какая из этих нейросетей справится лучше с детекцией студентов в аудитории.

В реальных условиях детектировать лицо оказалось более эффективно, чем полностью человека, поскольку студенты располагаются за партами, а иногда и за компьютерами. В таких аудиториях YOLOv5 показывала неудовлетворительные результаты, не детектировала студентов на задних партах. MTCNN в свою очередь работала лучше и могла определять студента даже по половине лица или в маске. Пример работы нейросети MTCNN показан на рис. 3.

Для более сложных условий, такие как: плохое освещение, перспективное искажение фотографии и её низкое качество, у MTCNN был путем экспериментов был настроен порог уверенности (thresholds = [0.7, 0.6, 0.6], дефолтные значения [0.6, 0.7, 0.7]), а также фотографии подавались на вход без сжатия. Это помогло немного улучшить качество детекции, однако, проблемные кейсы все равно остаются (чаще всего - детекция на задних партах в условиях плохой освещенности и низкого качества фотографии), в таких случаях преподавателю необходимо «досчитать» студентов самостоятельно.

Однако все еще остается шанс возникновения проблемы, когда один человек пришел, но не отметился, а другой отметился, но не пришел (такое возможно в связи с погрешностью определения геолокации). Исключить такие случаи помогает открытость баллов и информации о посещении для студента, то есть он видит, засчитано ли ему посещение и получил ли он баллы за это, следовательно не возникает ситуаций, когда студент пришел, но не отметился. Данное решение избавляет преподавателя от проблем с учетом посещаемости и позволяет ему сосредоточиться на проведении занятия. И даже при возникновении каких-либо вопросов со стороны обучающихся, объем взаимодействия сокращается до 1-2 студентов.

Рис. 3. Результат детекции нейросетью MTCNN

5. Учет посещаемости в онлайн формате

При дистанционном обучении все описанные выше методы перестают работать, ведь взаимодействие между преподавателем и студентом во время пары происходит посредством видеоконференции. Очевидным способом учета является перенос списка участников конференции в журнал посещаемости. Однако то, что студент подключился к занятию, еще не значит, что он на нем присутствует.

Поэтому в ЦОП Мирера автоматически оценивается в баллах активность участника видеоконференции и факт посещения фиксируется только при наборе участником количества баллов выше некоторого порогового значения. Кроме того, за высокую активность студенту могут быть начислены дополнительные баллы, которые могут учитываться при выставлении оценки за курс. Технологически эта идея реализована так. В ЦОП Мирера для проведения видеоконференций используется система BigBlueButton [11], из которой можно получить информацию не только о самом факте участия человека в видеоконференции, но информацию о его активности во время занятия (включенная камера, устные ответы, ответы в чате). Также может учитываться время просмотра записи конференции, которое реализовано путем встраивания видеопроигрывателя в систему, а также подсчетом времени, в течение которого у студента проигрыватель был открыт. Каждый раз, когда студент сворачивает видео, перезагружает или уходит со страницы, информация о времени просмотра передается на сервер. Однако все еще остается проблема того, что студент может несколько раз пересматривать один и тот же фрагмент или открыть видео, но область видимости в браузере не будет покрывать видеопроигрыватель (например, если студент пролистал страницу ниже).

За каждый пункт активности студент автоматически получает дополнительные баллы. Причем баллы ставятся пропорционально в зависимости от границ, которые установил преподаватель, что стимулирует студентов проявлять большую вовлеченность. Преподаватель же в свою очередь имеет удобный интерфейс в виде статистики, где отображены все данные об активности студента во время конференции (рис. 4).

Также педагог имеет возможность ставить или убирать дополнительные баллы обучающемуся за какую-то особенную деятельность на занятии и писать комментарий к данному посещению (рис. 5).

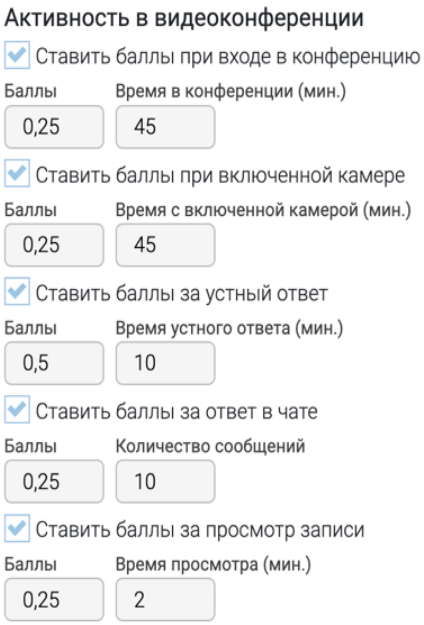

Рис. 4. Настройки баллов за активность

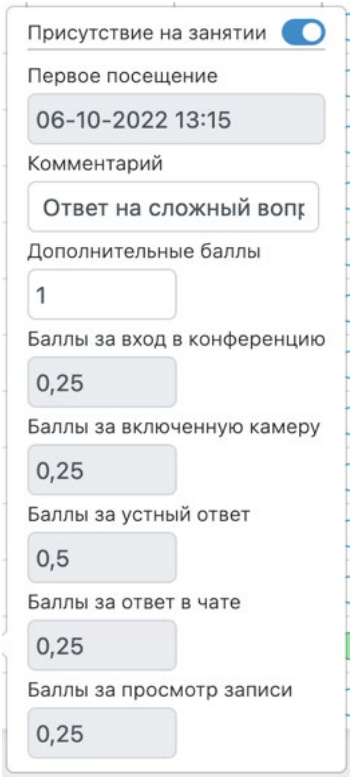

Рис. 5. Отчет об активности студента

6. Заключение

Учет посещаемости становится гораздо проще при использовании цифровых образовательных платформ. Методы, описанные в данной статье, позволяют пользователям (студентам и преподавателям) не отвлекаться от основной деятельности и полностью посвятить себя процессу обучения. Так, к примеру, при использовании геопозиционирования затраты на взаимодействие преподавателя с учебной группой по поводу посещаемости сводится к минимуму, а в сочетании с автоматизированным подсчетом людей в аудитории эти затраты уменьшаются до нуля. Студент также практически не тратит время на регистрацию на данном занятии (контесте) и в процессе регистрации мгновенно получает подтверждение того, что регистрация прошла успешно. Это исключает психологически нагрузочные ситуации, когда студент уверен, что он зарегистрировался на данном занятии, а фактически регистрация не произошла. Учет посещаемости в дистанционном формате позволяет не только отметить всех присутствующих, но и учесть их активность, что в дальнейшем позволит преподавателю корректировать свои занятия, а студентов мотивирует к большей вовлеченности, ведь баллы, полученные за посещения, также влияют на итоговый результат студента по курсу.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0010.

Approaches to Recording Student Attendance in the Digital Educational Platform of Mirera

A.G. Leonov, K.A. Mashchenko, A.V. Shlyakhov, A.A. Kholkina

Abstract. The interaction between a teacher and a student reaches a new level of collaboration when using educational platforms: learning becomes faster and more efficient. However, at the same time, many routine processes remain that make the work of the teacher difficult, such as, for example, recording student attendance. Attendance control should be carried out by the teacher, regardless of whether the lesson is held in person or remotely. The solution to this problem should have a certain convenience both for the teacher, minimizing the time spent on registering those present at the lesson, and for the students themselves. The article proposes a solution to this problem on the example of Mirer's author's digital educational platform. The possibility of using such a mechanism to increase the involvement of students in the learning process is noted.

Keywords: digital educational platform, digital educational environment, Mirera, attendance, videoconference, geolocation, neural networks

Литература

1. Леонов А.Г., Первин Ю.А. Качественные оценки эффективности методики обучения элементам информатики в пропедевтическом курсе // Ярославский педагогический вестник, №5, с. 92-96

2. Григорьев, П. В. Особенности технологии RFID и ее применение [Электронный ресурс] URL: https://moluch.ru/archive/115/30692/ (дата обращения: 1.11.2022).

3. Иванова Е.В., Струева А.Ю. Система учета посещаемости на основе распознавания лиц // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021 Т. 10, № 4 С. 60–73. DOI: 10.14529/cmse210404.Seth Gilbert, Nancy Lynch, Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services, ACM SIGACT News, Volume 33, Issue 2, June 2002

4. Бесшапошников Н.О., Дьяченко М.С., Леонов А.Г., Мащенко К.А. Использование элементов искусственного интеллекта в виртуальных помощниках преподавателя // Успехи кибернетики, 2020, том 1, №3, с. 15-22

5. Платформа Мирера [Электронный ресурс] URL: https://www.mirera.ru (дата обращения: 01.11.2022)

6. Geolocation API [Электронный ресурс] URL: https://developer.mozilla.org/en-US/docs/Web/API/Geolocation_API/ (дата обращения: 01.11.2022)

7. O. Russakovsky ImageNet Large Scale Visual Recognition Challenge // IJCV – 2015. – Vol. 115, No 3. – P. 211–252.

8. YOLOv5 [Электронный ресурс] URL: https://github.com/ultralytics/yolov5/wiki/ (дата обращения: 01.11.2022)

9. MTCNN [Электронный ресурс] URL: https://github.com/timesler/facenet-pytorch/ (дата обращения: 01.11.2022)

10. Задача нахождения объектов на изображении [Электронный ресурс] URL: https://neerc.ifmo.ru/wiki/index.php?title=Задача_нахождения_объектов_на_изображении (дата обращения: 01.11.2022)

11. BigBlueButton [Электронный ресурс] URL: https://bigbluebutton.org/ (дата обращения: 01.11.2022).

Реализация нового типа задач «электронные таблицы» в цифровой образовательной платформе Мирера

И.А. Васильев[1], А.Г. Леонов[2], К.А. Мащенко[3], А.В. Шляхов[4]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, vanya71161@gmail.com;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kirill010399@vip.niisi.ru;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, shlyakhov@vip.niisi.ru

Аннотация. Электронные таблицы успешно используются во многих областях экономики, финансах, бизнесе и т.п. Для получение школьниками и студентами компетенций по работе с электронными таблицами было принято решение добавить в цифровую образовательную платформу Мирера возможность использовать в курсах задания с электронными таблицами, для чего и был создан соответствующий авторский модуль. Изучение электронных таблиц определено ФГОСом по информатике основной школы, задачи на использование электронных таблиц входят в государственные итоговые испытания школьников. Разработанный модуль полностью покрывает требуемый объем знакомства с электронными таблицами и позволяет создавать контесты с автоматизированной проверкой в цифровой образовательной платформе Мирера, не устанавливая на компьютер ученика какое-либо дополнительное программное обеспечение.

Ключевые слова: информатика, цифровая образовательная платформа Мирера, электронные таблицы, работа с данными, формулы

1. Введение

Бум активного использование МООК систем, который произошел во втором десятилетии XXI века, не сопровождался предоставлением педагогам полного набора автоматизированных ресурсов для проверки заданий студентов [1]. В российском образовании уделяется определенное внимание облегчению труда педагога путем автоматизации рутинных функций по проверке заданий [2]. Авторы уверены, что цифровизация образования может сделать преподавание естественно-научных дисциплин более эффективным за счет включения в каждый классический курс большого набора практических заданий, проверка правильности выполнения которых обучаемым автоматически выполняется компьютерной системой [3]. Разработанная авторами цифровая образовательная платформа Мирера полностью отвечает сформулированному критерию [4]. ЦОП Мирера предлагает широкий круг тестирующих систем по различным дисциплинам, от компиляторов до автоматической оценки текстового задания [5], и ее возможности расширены внедрением электронных таблиц [6].

Популярность и широкое распространение электронные таблицы получили еще в прошлом веке. Для того, чтобы уметь обрабатывать табличные данные, производить на их основе расчеты, а также успешно сдать единый государственный экзамен [7], включающий в себя обработку таблицам [8], современным школьникам и студентам необходимо иметь определенные компетенции по работе с электронными таблицами, для чего в цифровую образовательную платформу Мирера был интегрирован новый тип задач «электронные таблицы». Студентам и преподавателям предлагается весь необходимый функционал, требующийся для решения различных задач, так или иначе связанных с обработкой табличных данных: автоматическая проверка решений, шаблоны в задачах с возможностью запрета редактирования и использование базовых формул. В отличие от цифровых образовательных сред КуМир [9] и ПиктоМир [10], которые являются самостоятельными устанавливаемыми приложениями, модуль электронных таблиц полностью интегрирован в Мireру, поэтому для выполнения заданий по электронным таблицам в Мирере не требуется устанавливать на компьютер ученика никакие дополнительные программные системы.

2. Механизм шаблонов

В новом типе задач «электронные таблицы», как и во всех других типах задач, используемых в Мирере, реализована метафора шаблона – за-

ранее подготовленной преподавателем защищенной части решения. Наличие шаблонов является серьезной помощью студенту. Однако, если, например, в типе задач «программирование» шаблон нужен студенту для того, чтобы, отталкиваясь от него, написать свой собственный код, то в задачах на таблицы преподаватель может предоставлять в качестве шаблона данные, с которыми нужно производить вычисления. Предоставляемые данные можно защитить от редактирования студентом, чтобы исключить возможность случайно их испортить. На рис. 1 представлен пример шаблона в таблице у студента.

Интерфейс шаблона в электронных таблицах фактически точно такой же, как и в остальных типах задач: ячейки, принадлежащие шаблону, окрашиваются в темно-серый цвет и становятся заблокированными от редактирования. Реализован данный функционал путем хранения массива ячеек, принадлежащих шаблону. Таким образом, всегда, когда студент будет открывать свое решение, будет осуществляться пробег по таблице и замена ячеек, принадлежащих этому массиву, на ячейки шаблона.

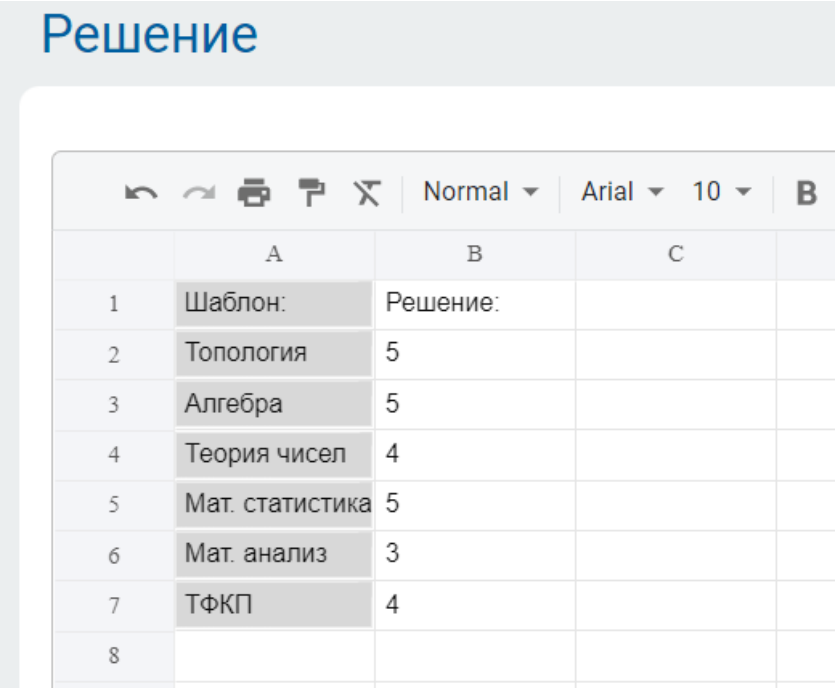

Рис. 1. Шаблон в задаче

3. Формулы

Поскольку одним из основных назначений нового типа задач «электронные таблицы» было проведение тестирований в формате ЕГЭ (единого государственного экзамена) по информатике на основе цифровой образовательной платформы Мирера, очень важной частью проделанной работы является реализация механизма формул. Задачи экзамена заключаются в обработке данных, приведенных в условии: подсчету средней температуры, поиска ее максимума или минимума за определенный срок, сортировке столбцов и строк, и так далее. Для выполнения подобного рода задач, необходима реализация достаточно полного набора формул, который в текущей реализации состоит из десятка формул: СУММ, МАКС, МИН, КОНКАТ, ЕСЛИ, И, ИЛИ, СЧЕТЕСЛИ, СУММЕСЛИ, СРЗНАЧЕСЛИ, имена которых можно писать и на английском. (В примерах ниже мы используем английскую транскрипцию).

Номенклатура формула была выбрана так, чтобы в электронных таблицах Мирэры могли быть решены все задания последних лет ЕГЭ по информатике, подразумевающие использование электронных таблиц. Основная сложность в реализации формул заключалась в том, чтобы сделать максимально информативными и понятными сообщения об ошибках в формулах.

На рис. 2 представлен пример работы формулы СУММЕСЛИ. Для того, чтобы показать и запись формулы, и ее результат, она введена дважды: слева – неактивное состояние (результат работы), справа – состояние редактирования (запись формулы).

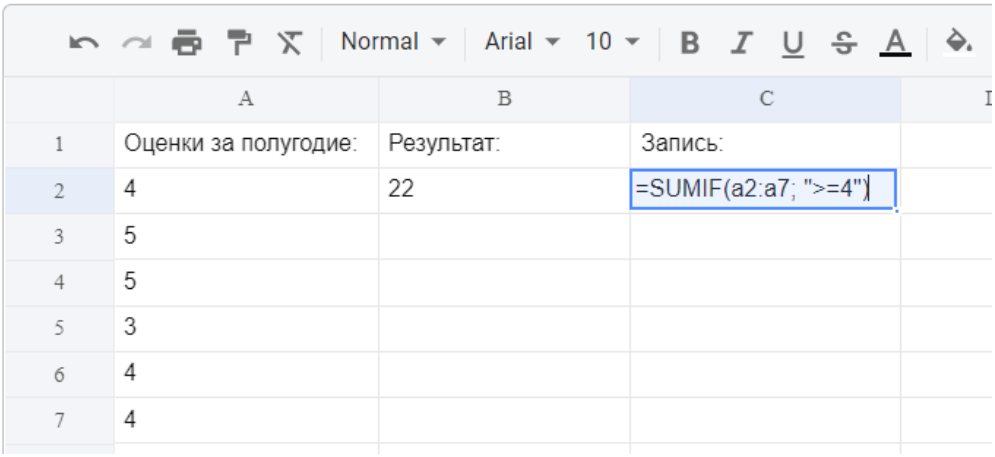

Рис. 2. Пример работы формулы SUMIF

3.1. Реализация вложенности формул

Хотя при решении задач ЕГЭ необходимость вложений формул с большой глубиной не встречается, было решено реализовать вложенность произвольной глубины.

Для реализации вложенности было использовано некоторое подобие рекурсивной обработки введенной в ячейку строки. Во время ее парсинга мы можем неоднократно встретить символ “(“, причем первое его появление свидетельствует лишь о вызове обычной формулы, а вот второе и следующие - о том, что в ячейку введено несколько формул. Для обработки такого случая был заведен счетчик открывающихся скобок, и как только его значение начинает превышать единицу, функция, обрабатывающая ячейку, вызывается снова внутри себя самой и в качестве аргумента получает уже лишь некоторую подстроку, содержащую вложенную формулу. После того, как дочерняя функция отрабатывает, родительская получает ее результат, то есть, результат работы подформулы, с которым теперь уже тоже можно работать как с обычном числом или строкой. Конечно, таких вложенностей может быть и несколько, работа функций от этого никак не изменится, просто увеличится количество их вызова внутри себя. На рис. 3 представлен пример работы вложенных формул =ЕСЛИ() и =ИЛИ(), с вызовом второй внутри первой.

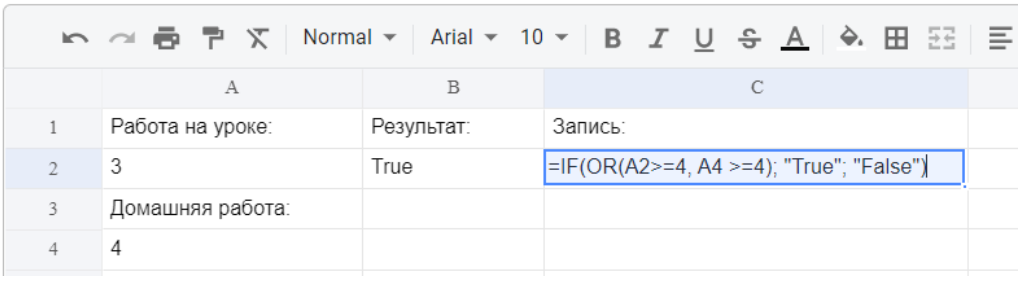

Рис. 3. Вложенные формулы IF и OR

4. Система хранения и рендеринга данных

Для того, чтобы интерфейс электронных таблиц не сильно отличался привычного в России интерфейса зарубежных прототипов, была реализована достаточно сложная система хранения и отрисовки данных таблиц. На сервере вся информация о решении студента хранится в виде файла, распаковывая который можно получить JSON строку, парсинг которой приводит нас к объекту, хранящему в своих полях полную информацию о решении. Для последующей отрисовки числовых, строковых или других данных, нам всего лишь нужно обратиться соответствующим образом к объекту и отрисовать полученные данные. Однако, если, например, в ячейку таблицы была введена формула вида =СУММ(A1:A10), то нам нужно рендерить именно результат работы формулы, а не ее запись, и только при непосредственном двойном нажатии на эту ячейку показывать запись фор-

мулы. Для этого было реализовано несколько последовательно вызывающихся функций, которые анализируют содержимое переданной им в аргумент ячейки и на основе полученной информации возвращают финальное значение, которое необходимо отобразить в таблице. В качестве примера на рис. 4 изображены две одинаковые формулы, причем с левой стороны – результат работы формулы (то есть неактивное состояние ячейки), а с правой – формальная запись формулы (то есть ее состояние после двойного нажатия на ячейку).

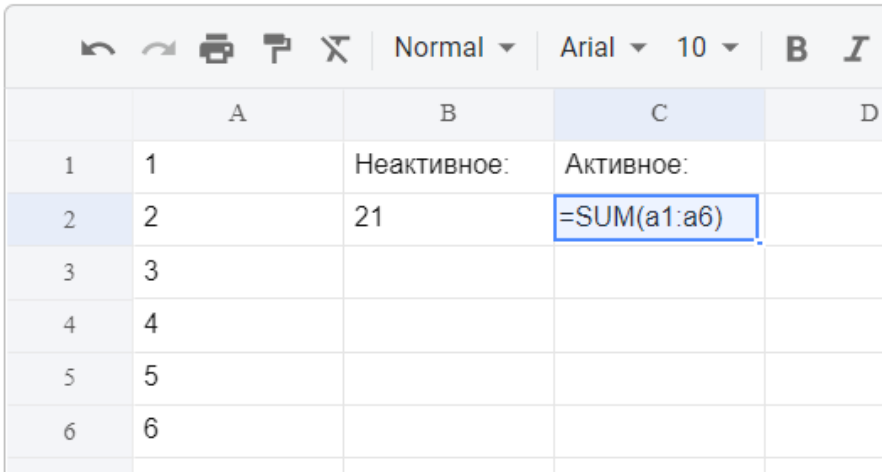

Рис. 4. Активное и неактивное состояние ячейки с формулой

5. Проверка решений

После того, как был реализован основной функционал электронных таблиц, оставался последний шаг – внедрение такой же автоматической проверки, которая используется в других типах задач. Процесс проверки осуществляется следующим образом: в функцию проверки приходят два файла - решение студента и эталонное решение, после чего в цикле сверяются все ячейки обеих таблиц. Важно отметить, что ячейки проверяются по-разному в зависимости от их содержимого. Если ячейки содержат обычные данные, то для прохождения проверки им необходимо быть полностью идентичными. Если же ячейка начинается с символа “=”, т.е. содержит формулу, то перед проверкой из нее удаляются все пробельные символы, т.к. они не играют никакой роли и наличие лишнего пробела в формуле у студента не должно являться ошибкой. Другой важной особенностью является то, что таблицы проверяются “в обе стороны”, т.е. все непустые ячейки, содержащиеся в эталонном решении, должны быть также и в решении студента, и наоборот.

6. Примеры использования электронных таблиц

Разберем в качестве примера создание и решение типового задания №9 из ЕГЭ по информатике [11]. По условию нам дан файл с электронной таблицей, содержащей данные о тестировании учеников по выбранным ими предметам. В столбце A записан код округа, в котором учится ученик; в столбце B – фамилия, в столбце C – выбранный учеником предмет; в столбце D – тестовый балл. Всего в электронную таблицу были занесены данные по 1000 учеников.

Необходимо определить количество учеников, проходивших тестирование по информатике и набравших более 600 баллов, и записать ответ в ячейку H2. А также найти средний тестовый балл учеников, проходивших тестирование по информатике, ответ записать в ячейку H3. Для того, чтобы создать такую задачу на цифровой образовательной платформе Мирера, преподаватель должен проделать следующую последовательность действий:

1. Скопировать условие в описание задачи в контесте.

2. Скачать таблицу, данную в задаче, и вставить ее в эталонное решение.

3. Для ответа на первый вопрос необходимо записать в ячейку E2 формулу =ЕСЛИ(И(D2>600; C2="информатика");D2;0) и скопировать ее в диапазон E3:E1001.

4. Применить операцию =ЕСЛИ(E2>0;1;0) и получить столбец (F): с единицами и нулями. Далее, использовать операцию =СУММ(F2:F1001).

5. Для ответа на второй вопрос необходимо записать в ячейку G2 следующее выражение: =ЕСЛИ(C2="информатика"; D2;0), в результате применения данной операции к диапазону ячеек

G2:G1001, получить столбец, в котором записаны баллы только учеников, сдававших информатику.

6. Сложить значения в ячейках, получить сумму баллов учеников. Найти количество учеников, сдававших информатику, с помощью команды =СЧЁТЕСЛИ(C2:C1001;"информатика") и разделить полученное значение на количество учеников.

7. В шаблоне нажать кнопку “Скопировать эталонное решение”, после чего из шаблона стереть введенные ранее формулы.

8. Выключить кнопку “Разрешить студенту редактирование”.

После выполнения этих простых шагов, задача будет полностью готова, и студенты смогут приступить к ее выполнению. На рис. 5 представлен заблокированный шаблон с данными из описанной выше задачи.

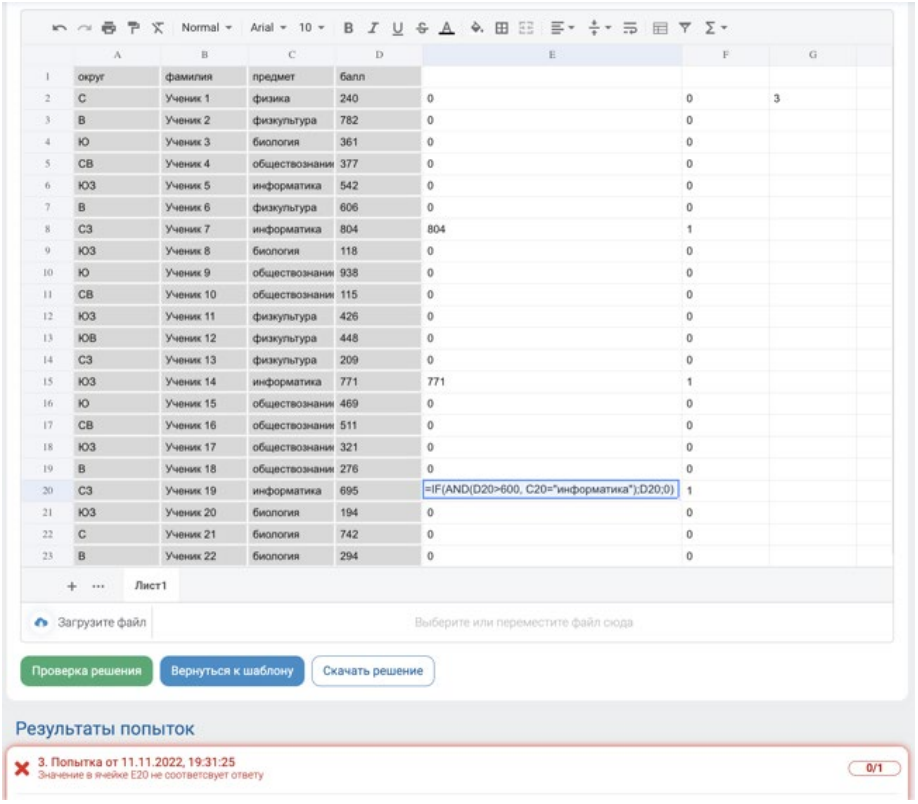

Рис. 5. Пример решения задачи из государственного экзамена

7. Заключение

Новый встроенный модуль электронных таблиц расширил возможности цифровой образовательной программы Мирера, позволив проводить на ее основе этапы единого государственного экзамена, а также различные расчеты, связанные с данными из таблиц.

В дальнейшем планируется еще большее расширение списка базовых формул, которое бы позволило производить более сложные вычисления с электронными таблицами. Жизненно необходимым также является расширение возможности автоматической проверки, которые позволили бы, например, признавать равными ячейки решения студента и эталонного решения, которые содержат в себе разные формулы, дающие одинаковый результат.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0010.

Implementation of a New Type of Tasks «Spreadsheets» in the Digital Educational Platform of Mirera

I.A. Vasilyev, A.G. Leonov, K.A. Mashchenko, A.V. Shlyakhov

Abstract. Spreadsheets are successfully used in many areas of the economy, finance, business, etc. In order for schoolchildren and students to gain competencies in working with spreadsheets, it was decided to add the ability to use spreadsheet tasks in courses to Mirera's digital educational platform, for which a corresponding author's module was created. The study of spreadsheets is defined by the Federal State Educational Standard for Informatics of the main school, tasks for the use of spreadsheets are included in the state final tests of schoolchildren. The developed module fully covers the required amount of familiarity with spreadsheets and allows you to create contests in automated verification in Mirera's digital educational platform without installing any additional software on the student's computer.

Keywords: informatics, digital educational platform Mirera, spreadsheets, data processing

Литература

1. F. Grünewald, C. Meinel, M. Totschnig, C. Willems. Designing MOOCs for the Support of Multiple Learning Styles. In: Hernández-Leo, D., Ley, T., Klamma, R., Harrer, A. (eds) Scaling up Learning for Sustained Impact. EC-TEL 2013. Lecture Notes in Computer Science, vol 8095. Springer, Berlin, Heidelberg.

2. Ф. Л. Косицкая Основные тренды в современном российском высшем образовании (по материалам зимней школы преподавателей – 2020). Ped.Rev.. 2020. №3 (31). URL: https://cyberleninka.ru/article/n/osnovnye-trendy-v-sovremennom-rossiyskom-vysshem-obrazovanii-po-materialam-zimney-shkoly-prepodavateley-2020 (дата обращения: 01.09.2022).

3. А.Г. Леонов, Н.О. Бесшапошников, А.А. Прилипко. Цифровизация образования – Новые возможности управления образовательными треками. Вестник кибернетики. – 2018. – Т. 30, № 3. – С. 154–161.

4. Платформа Мирера [Электронный ресурс] URL: https://www.mirera.ru (дата обращения: 01.09.2022)

5. А.Г. Кушниренко, М.А. Кузьменко, А.Г. Леонов. Элементы цифровизации образовательного процесса на примере системы Мирера. Свободное программное обеспечение в высшей школе : сб. материалов 13-й конф. М. : BaseaIt, 2018. С. 66–68.

6. А. Г. Леонов, М. С. Дьяченко, К. А. Мащенко и др. Новые подходы к автоматизации проверки заданий в цифровых курсах. Информатизация образования и методика электронного обучения: цифровые технологии в образовании Материалы VI Международной научной конференции. – Красноярский государственный педагогический университет им. В.П. Астафьева, Красноярск, 2022. – С. 173–179.

7. Примерная рабочая программа основного общего образования информатика. Базовый уровень (для 7-9 классов образовательных организаций), Одобрена решением федерального учебно-методического объединения по общему образованию, протокол 3/21 от 27.09.2021 г.Москва, 2021.

8. С.С. Крылов. Методические рекомендации для учителей, подготовленные на основе анализа типичных ошибок участников ЕГЭ 2021 года по информатике и ИКТ. Педагогические измерения. ФГБНУ «Федеральный институт педагогических измерений», том 4. Москва 2022 с.29-46, 2021.

9. Стартовая страница проекта «КуМир» на сайте ФГУ ФНЦ НИИСИ РАН. [Электронный ресурс] URL: https://www.niisi.ru/kumir (дата обращения: 01.09.2022).

10. Стартовая страница проекта «ПиктоМир» на сайте ФГУ ФНЦ НИИСИ РАН. [Электронный ресурс] URL: https://www.niisi.ru/piktomir (дата обращения: 01.09.2022).

11. Образовательный портал для подготовки к экзаменам РЕШУ ЕГЭ https://inf-ege.sdamgia.ru/ (дата обращения: 01.09.2022).

Визуализация алгоритмов реализации взаимоисключений средствами пиктограммной среды программирования

И.Н.Грибанова[1], А.С. Караваева[2], А.А. Леонов[3], А.Г. Леонов[4], Д.В. Мащенко[5], В.А. Оганисян[6]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, nig@niisi.com;

[2]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, aleksandrakaravaevaa@yandex.ru;

[3]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, anton11-02@mail.com;

[4]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, МГУ им. М. В. Ломоносова, Москва, Россия, МПГУ, Москва, Россия, Государственный университет управления, Москва, Россия, dr.l@vip.niisi.ru;

[5]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, mashchenko.darya.vlad@yandex.ru;

[6]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, kovyrshina@niisi.ru

Аннотация. Курсы по информатике и информационно-коммуникационным технологиям могут включать широкий набор тем от основ программирования и элементарных знаний об устройстве компьютеров до теории формальных грамматик, языков искусственного интеллекта, операционных систем, 3D-графики и т.п. Основную сложность при этом составляет математическая компонента курсов, что требует соответствующей подготовки слушателей. Однако, и в программировании есть темы, сложные для понимания студентами даже с глубокими академическими знаниями. Одной из таких тем является параллельное программирование, когда с одной стороны требуется изучить и написать учебные высокоэффективные алгоритмы решения задач для выполнения одновременных вычислений на различных ЭВМ или процессорах, с другой стороны, что не менее сложно для понимания - обеспечить параллельную работу различных частей программы с разделяемым доступом к общим данным. Освоение азов параллельного программирования усложняется отсутствием адекватной визуализации параллельной работы процессов на компьютере. Педагогам остается иллюстрировать тему описывая знакомые студентам ситуации из реального мира, например, задачу одновременного проезда четырех автомобилей через нерегулируемый перекресток. С другой стороны, современные тенденции требует понижения возраста начала знакомства растущего ребенка с элементами программирования и алгоритмизации. Дошкольники, дети четвертого года жизни, с успехом осваивают в игре основные понятия последовательного программирования. используя программно-управляемых роботов и их виртуальных двойников в цифровой образовательной среде ПиктоМир. Та же ЦОС ПиктоМир используется для пропедевтики программирования при преподавании для любого возраста, включая студентов университетов. ПиктоМир обладает не только простотой в освоении, но допускает использование в заданиях одновременно нескольких исполнителей-роботов, каждый из которых действует по собственной программе. Наглядность процесса выполнения программ в ПиктоМире привела авторов к мысли использовать ЦОС ПиктоМир для демонстрации работы классических алгоритмов взаимоисключений в курсах по параллельному программированию. Опыт авторов показал высокую эффективность подобного подхода, что позволило студентам за короткий срок понять проблему и успешно решить все задачи темы. Указанный подход апробировался в курсах Операционные системы (ИИС ГУУ) и Алгоритмы и структуры данных (Институт Детства МПГУ).

Ключевые слова: взаимное исключение, процесс, deadlock, критическая секция, ПиктоМир. операционные системы, алгоритмы.

1. Введение

Вопросы параллельных вычислений стали активно исследоваться с 60-х годов. Необходимость таких работ легко поясняется примерами того времени. В СССР, например, в Москве в больших киноконцертных залах всегда было несколько касс. В случае больших загрузки, с целью исключить большие очереди на текущий киносеанс, билеты продавались сразу в двух и более кассах. В отсутствии компьютеров и информационных систем, кассиры должны были договариваться между собой, кто какие места продает. В случае, если билеты не содержали отпечатанный типографским способом ряд и место в зале, кассирам приходилось отмечать проданные места на одном листе схемы киноконцертного зала, но и даже в случае, когда билет содержал напечатанное место и ряд создавались определенные сложности. Так, например, пара кассиров делила пачки билетов примерно пополам, и

каждая касса продавала свою часть. Кассиры работали с разной скоростью, поэтому в процессе продажи у одной из касс могла выстроиться очередь, а другая вообще оказывалась свободной в случае распродажи всех отведенных кассе билетов. Аналогичные трудности возникали и при продаже билетов в железнодорожных кассах дальнего следования. Если касса не находилась непосредственно на вокзале, то кассир вынужден был заранее получать список билетов, которые он может реализовать, или ему требовалось регулярно дозваниваться до вокзала, сообщая в центральную кассу о факте продаже билета на конкретный поезд или спрашивать о наличии места на данное направление. Естественно, что такая работа протекала чрезвычайно медленно, создавая очереди, часто возникали сбои, когда, например, несколько билетов на одно и то же место продавали одновременно разным пассажирам. Можно говорить, что в информационной модели системы по продаже билетов, кассиры и выполняемая ими работа представляли независимые процессы, а список билетов – общий ресурс. При этом работа процессов – касс была организована параллельно, и если не учитывать общий ресурс, то независимо друг от друга.

Задачу синхронизации параллельной работы процессов можно решить и для N процессов (или ЭВМ), связанных между собой только обращением к общему ресурсу [1]. Для этого процесс рассматривается как отдельная программа, в которой выделяется часть, называемая, обычно, критической секцией или критическим интервалом. В этом месте программы осуществляется доступ к общим данным, который должен быть монопольным, то есть одновременно в своей критической секции может находиться только один процесс.

Считается, что первой статьей, решающей эту проблему, называемую проблемой взаимного исключения, была статья Эдсгера Дейкстры [2], в которой автор сформулировал условия задачи обеспечения взаимоисключающего доступ к критической секции среди конкурирующих процессов.

- каждый процесс выполняет последовательность инструкций (команд) в бесконечном цикле.
- инструкции разделены на четыре части: код остаточная часть (не критично одновременное выполнение), вход в критическую секцию (код, выполняющийся перед входом в критический интервал), критический интервал и выход (код, который выполняется после критического интервала).

Для решения поставленной задачи необходимо написать программный код для двух разделов входа и выхода, чтобы удовлетворялось условие, что никакие два процесса не находятся в своих критических секциях одновременно. Предполагается, что код критической секции и остальной части процесса не будет меняться при обеспечении взаимоисключений. Естественно также предположить, что решение будет идентичным для всех процессов и будет использовать общие переменные [3].
Исходный код каждого процесса выглядит так:

```
loop forever
    remainder code;
    entry code;
    critical section;
    exit code;
end loop
```

На рис. 1 представлено решение Эдсгера Дейкстры из указанной выше статьи.

В дальнейшем алгоритм неоднократно улучшался. Так в 1974 Лесли Лампортом был предложен так называемый алгоритм Пекарни [4], работающий по аналогии с настоящим заведением. Покупатели-процессы, заходят в пекарню, и каждый входящий по очереди получает номерок, то есть на единицу больше предыдущего покупателя. Специальное табло показывает номер клиента, обслуживаемого в данный момент времени. Остальные посетители ожидают в очереди, пока не закончится обслуживание клиента и табло не покажет следующий номер. Клиент, который уже совершил покупку, отдает свой номерок продавцу и тот не только вызывает следующего по очереди, но и увеличивает на единицу допустимое число входных номеров. Покупатели-процессы получают номера из некоторой глобальной величины [5].

Алгоритм неоднократно дорабатывался, из-за проблем с одновременным получением несколькими процессами одного и того же номера [6,7].

Несмотря на наличие большого количества статей по данной тематике представленные алгоритмы или не удовлетворяли принципу очередности (FIFO), либо использовали неограниченные целые величины. Очередное решение было предложено уже в этом веке [8]. Автор утверждает, что все требования проблемы взаимоисключений были выполнены.

Возвращаясь к работам Эдсгера Дейкстры отметим, что он не только опубликовал первым статью в области параллельных вычислений, выявившей и решившей проблему взаимного исключения, но и много сделал в области самих распределённых вычислений, а также ввел понятие семафоров (semaphore) - целочисленных ве-

личин с атомарными операциями как примитивов синхронизации процессов (потоков).

```
“integer j;
Li0:  b[i] := false;
Li1:  if k ≠ i then
Li2:  begin c[i] := true;
Li3:  if b[k] then k := i;
      go to Li1
      end
        else
Li4:  begin c[i] := false;
        for j := 1 step 1 until N do
          if j ≠ i and not c[j] then go to Li1
      end;
      critical section;
      c[i] := true;  b[i] := true;
      remainder of the cycle in which stopping is allowed;
      go to Li0”
```

Рис. 1. Алгоритм взаимоисключений для N процессов Э.Дейкстры.

2. Классические задачи на алгоритмы взаимоисключений

Для облегчения понимания алгоритмов взаимоисключений проще рассматривать проблему на примере двух процессов, используя для синхронизации простые переменные [9].

Более того, для школьников естественно использовать для объяснения алгоритмы, реализованные на школьном алгоритмическом языке, который также часто называют КуМир (по имени программной среды, в которой этот яхык реализован)[10].

Здесь задача представлена в переформулированном виде: требуется решить проблему синхронизации выполнения двух процессов [11], где алгоритм-процесс выполняется циклически, вначале выполняется вспомогательный алгоритм «критическая секция», а затем выполняется «окончание алгоритма», где процессу уже не требуется общий ресурс:

```
алг процесс

нач

| нц

| | критическая секция

| | окончание алгоритма

| кц

кон
```

Естественно, что это еще не решение проблемы, поскольку отсутствует синхронизация. Модифицируем алгоритмы, так, чтобы свою критическую секцию мог выполнять один и только один процесс, что и будет решением проблемы синхронизации. Это решение состоит в добавлении целой переменной «переключатель» с двумя значениями 1 и 2, по номерам процессов, которая принимает эти значения, в зависимости от номера процесса, допущенных в данное время в свою критическую секцию.

На рис. 2 представлено решение проблемы взаимоисключений с переключателем (КуМир). Считается, что алгоритмы в нижних прямоугольниках выполняются параллельно.

Пока переключатель равен 1, процесс2 не может войти свою критическую секцию и находится в бесконечном цикле ожидания, пока переключатель не станет равен 2. В свою очередь переключатель станет равен 2, когда первый процесс выйдет из критической секции, и изменит содержимое переключателя. Так же работает и второй процесс.

Как и ожидалось, в каждый момент времени только один процесс может находится в своем критическом интервале. Но процессы входят в

свои критические секции строго последовательно: первый, второй, первый, второй...

При таком подходе один из процессов может быть неоправданно задержан, хотя общий ресурс будет свободен для использования.

Здесь, естественно, нужно сформулировать второе правило синхронизации: задержка любого процесса, вне его критического интервала не должна влиять на ход работы остальных процессов.

На рис. 3 представлено решение проблемы взаимоисключений с двумя флагами (КуМир).

```
алг параллельные процессы
нач цел переключатель
| переключатель := 1
| процесс1 : процесс 2 | параллельное выполнение процессов
кон
```

```
алг процесс 1
нач
| нц
| | нц пока переключатель = 2
| | кц
| | критическая секция1
| | переключатель := 2
| | окончание 1
| кц
кон
```

```
алг процесс 2
нач
| | нц
| | нц пока переключатель = 1
| | кц
| | критическая секция2
| | переключатель := 1
| | окончание 2
| кц
кон
```

Рис. 2. Решение проблемы взаимоисключений с переключателем (КуМир).

Интуитивно, представленный алгоритм с двумя флагами кажется более удачным: он не требует соблюдения четкой последовательности выполнения критических секций. Если один из процессов работает быстрее, то он сможет чаще попадать в критическую секцию. Но и этот алгоритм не лишен недостатков: не исключен случай, когда оба процесса готовы войти в критические секции (флаг1 и флаг2 приняли значения «да»), тогда ожидание в цикле будет длиться бесконечно и не один из процессов не сможет выполнятся дальше. Эта проблема приводит к формулировке третьего требования: решение относительно того, какой процесс должен войти в свой критический интервал, должно быть принято за конечное время.

Как известно, классическое решение проблемы взаимоисключений требует использования одного переключателя и двух флагов.

На рис. 4 представлено решение проблемы взаимоисключений с тремя переменными (КуМир).

Это решение является фактическим объединением первого и второго вариантов. Для синхронизации требуются и переменная переключатель, и флаги (флаг1 и флаг2). Переключатель "отвечает" за номер процесса, который выполняется в текущий момент в критической секции. Можно заметить, что величина переключатель вообще не требует инициализации, так как устанавливается в 1 или 2 до цикла ожидания. В отличии от переключателя, флаги в начале сбрасываются (им присваивается «нет»). Их нельзя не инициализировать, так как какой-нибудь из процессов может «вырваться» вперед, и другой не успеет проинициализировать свой флаг. В начале процессов флаг поднимается (ему присваивается «да»), то есть процесс готов войти в критическую секцию. Переключателю присваивается номер другого процесса, чтобы «пропустить» на выполнение этот другой процесс. Цикл ожидания прервется, если тот, или другой процесс покинет свою критическую секцию (установив свой флаг в 0). Фактически функция переключателя состоит только в обходе ситуации, которая могла возникнуть на втором решении, то есть исключить взаимную блокировку процессов.

```
алг параллельные процессы
нач лог флаг1, флаг2
| флаг1 := "нет"
| флаг2 := "нет"
| процесс1 : процесс 2 | параллельное выполнение процессов
кон
```

```
алг процесс 1
нач
| нц
| | флаг1 := "да"
| | нц пока флаг2
| | кц
| | критическая секция1
| | флаг1 := "нет"
| | окончание 1
| кц
кон
```

```
алг процесс 2
нач
| нц
| | флаг2 := "да"
| | нц пока флаг1
| | кц
| | критическая секция2
| | флаг2 := "нет"
| | окончание 2
| кц
кон
```

Рис. 3. Решение проблемы взаимоисключений с двумя флагами (КуМир)

```
алг параллельные процессы
нач лог флаг1, флаг2, цел переключатель
| флаг1 := "нет"
| флаг2 := "нет"
| процесс1 : процесс 2 | параллельное выполнение процессов
кон
```

```
алг процесс 2
нач
| нц
| | флаг1 := "да"
| | переключатель := 2
| | нц пока флаг2 и
| | переключатель = 2
| | кц
| | критическая секция1
| | флаг1 := "нет"
| | окончание 1
| кц
кон
```

```
алг процесс 2
нач
| нц
| | флаг2 := "да"
| | переключатель := 1
| | нц пока флаг1 и
| | переключатель = 1
| | кц
| | критическая секция2
| | флаг2 := "нет"
| | окончание 2
| кц
кон
```

Рис. 4. Решение проблемы взаимоисключений с тремя переменными (КуМир)

3. Постановка проблемы

Если ставить перед собой цель объяснять вопросы синхронизации параллельных процессов наиболее просто, то использование при изложении языка системы КуМир вполне оправдано, однако, так как ЦОС КуМир не имеет возможности параллельного выполнения программ, то ученики не смогут попрактиковаться в решении задач на параллельные процессы. Алгоритмы, записанные на школьном алгоритмическим языке фактически эквивалентны записи на псевдокоде: их можно составлять, исправлять, обсуждать, но нельзя выполнить на компьютере, что является достаточно странным методическим приемом в 21 веке. Однако, использование школьного алгоритмическог языка как псевдокода вполне оправдано.

Вполне естественно использование в образовательном процессе производственного языка программирования, такого как C++. Библиотеки современных фреймворков, такие как Qt, имеют обширную поддержку синхронизации параллельных процессов-нитей (Thread) в своем арсенале [12]. Тем интереснее смоделировать синхронизацию работы двух и более процессов, для получения демонстрации, например, взаимной блокировки двух процессов, решение проблемы взаимоисключений и т.д.

Программа содержит в себе запуск N=2 (nThread=2) процессов (нитей). Каждый процесс имеет свой целочисленный идентификатор (id = 0 или 1), который является параметром функций:

```
void Threads::Critical_section(int id)
и
void Threads::Reminder_section(int id)
```

Для успешного моделирования параллельных процессов, в эти функции включена не только проверка единственности процесса в своей критической секции, но и определенная существенная временная задержка, так как даже если не использовать никакие способы синхронизации нитей, попадание двух процессов одновременно в "короткую" критическую секцию практически исключено. Напротив, реальные процессы в реальных задачах могут находится в критических интервалах достаточно длительное время.

Заготовка основной функции процесса выглядит так:

```
      {
// код писать сюда
   Critical_section(id);
// код писать сюда
   Reminder_section(id);
    }
```

Для полноценного решения задачи по мотивам алгоритма Э.Дейкстры студентам предлагается две использовать глобальные переменны:

```
static int swich=0;
static int key[2]= {0,0};
```

Где swich - переключатель процессов (0 и 1), написанный с пропущенной буквой t, чтобы отличить переменную от оператора C++, а массив key[] - суть флаги процессов.

Наблюдая за выводом на экран, можно увидеть все ожидаемые эффекты, такие как чередование работы процессов (при использовании только одного переключателя), взаимную блокировку процессов (при попытке синхронизации с использованием только флагов) и нормальное выполнение, (при полноценной реализации алгоритма взаимоисключений, когда один процесс чаще другого попадает в критическую секцию и нет аварийной ситуации), анализ вывода показывает также что два процесса одновременно не оказываются в критическом интервале.

Однако, новичкам в программировании сложно понимать методы класса Thread-ов, а студентам тяжело представить, что один и тот же код выполняется различными процессами параллельно и одновременно, хотя визуально все содержится в одном файле.

К другим проблемам можно отнести эффекты, возникающие при работе алгоритмов оптимизации компилятора C++.

Во втором примере (на рис. 3) уже излагался неправильный метод синхронизации процессов, когда некоторая последовательность выполнения программы приводила к взаимной блокировке процессов. На C++ (в фреймворке Qt) этот пример выглядит так:

```
key[id]=1; while(key[1-id]);
   Critical_section(id);
key[id]=0;
  Reminder_section(id);
```

Можно обратить внимание, что одновременное выполнение первой строки в нитях приводит к взаимной блокировке, однако, имеется неопределённость поведения некоторых компиляторов C++, которое приводит к интересным последствиям.

Легко заметить, что переменой key[] дважды присваивается значение в функции. Но, эта величина нигде не участвует в вычислениях. Поэтому код может быть соптимизирован компилятором, когда операции присвоения просто будут выброшены из функции.

В качестве примера на рис. 6 изображено решение задачи взаимоисключений с использованием директив mutex() на C++ (Qt).

```
const int MaxWait = 200;
const int MaxSleep = 500; // Максимальный sleep
const int MaxRun = 30; // Максимальный
const int nThreads = 2; // Число потоков

class Threads : public QThread {

private:  int id;

public:
    Threads(int i):
        id(i){}
    void Critical_section(int);
    void Reminder_section(int);
    void run(void) override;

};


static int cr;
static int alarm=0;
static int swich=0;
static int key[2]= {0,0};
static QMutex mutex;

void Threads::run(void)
    {   for (int i=0; i<MaxRun*(nThreads-id); i++)
      {
// код писать сюда
//    while();

       mutex.lock();
    Critical_section(id);
      mutex.unlock();
//    swich=1-swich;

// код писать сюда
    Reminder_section(id);
      }
    }


void Threads::Critical_section(int i)
{    if (cr++) { alarm++; qDebug() <<  " Critical section " << i << " alarm  "  ;}
 //   QTime time = QTime::currentTime();
      msleep(static_cast<unsigned int>(QRandomGenerator::global()->bounded(MaxWait)));
        cr--;
}

void Threads::Reminder_section(int i)
{
     qDebug() <<  " Process " << i ;
//     QTime time = QTime::currentTime();
     msleep(MaxSleep*static_cast<unsigned long>(i));
     msleep(static_cast<unsigned int>(QRandomGenerator::global()->bounded(MaxWait)));
}


int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    Threads *Thread[nThreads];

    for (int k = 0; k < nThreads; k++)
        {
            Thread[k] = new Threads(k);
            Thread[k]->start();
        }
    for (int k = 0; k < nThreads; k++)
        {
            Thread[k]->wait();
        }
    return alarm > 0 ? -1 : alarm;
}
```

Рис. 5. Решение задачи взаимоисключений с использованием директив mutex() на C++ (Qt)

4. Визуализация алгоритмов

В предыдущем параграфе было указано на сложности освоения темы студентами, при использовании алгоритмов, реализованных на производственном языке программирования С++. Конечно, возможен подход, когда используется специально-разработанная библиотека для эффектной графической визуализации, например столкновения роботов-процессов при ошибках синхронизации или бесконечное ожидание перед некоторым объектом, как демонстрация взаимной блокировки процессов. Однако и это не может гарантировать успех занятиям, так как сложность языка С++ никуда не исчезает. К сожалению, подобные курсы доступны только студентам университетов или очень увлеченным старшеклассникам.

С другой стороны, ЦОС ПиктоМир уже несколько лет активно используется в дошкольных образовательных организациях, а также качестве пропедевтики программирования в школах, кружках, в курсах различных образовательных учреждениях и университетах [13].

ЦОС ПиктоМир обладает продуманным интерфейсом для составления простейших программ детьми даже в раннем возрасте [14]. В отличии от С++ на освоение языка Пикто (пиктографического языка ЦОС ПиктоМир) уходит совсем мало времени. Если студенты университета по какой-то причине еще не знакомы с ПиктоМиром. то для проведения полноценного занятия по алгоритмам взаимоисключений по мотивам решений Э.Лейкстры достаточно пары академических часов [15].

В ПиктоМире в качестве исполнителей используются роботы, выполняющие работу и двигающиеся по клетчатому полю-космодрому. Робот, который будет программироваться при решении задач называется Вертун. У Вертуна всего 4 команды:

- вперед;
- налево;
- направо;
- закрасить.

В простейшем случае будут использоваться только 3 команды и будут составлены индивидуальные программы для каждого из роботов.

Вводными пояснениями объясняются три требования к синхронизации алгоритмов, при использовании ЦОС ПиктоМир.

Три правила робототехники:

- Правило 1. В каждый момент времени только один робот может попытаться занять поле или ресурс.
- Правило 2. Задержка любого робота в рамках другой работы не должна влиять на ход работы остальных роботов.
- Правило 3. Решение занять поле или ресурс должно быть принято роботом за конечное время.

Таким образом разделяемым ресурсом для роботов будет клетка поля. Действительно, если роботы сталкиваются при одновременном прохождении одной и той же клетки, то они ломаются, что полностью соответствует результату столкновения в реальном, а не виртуальном мире.

В качестве примера на рис. 7 изображено исходное состояние и программа для робота, не использующая синхронизацию.

В этой задаче роботы должны многократно закрашивать (чинить) "сломанную" клетку, при этом программа для робота, отмеченная зеленым кружком (при черно-белой печати просто кружком) выполняет стандартные действия:

- робот идет два шага вперед
- красит клетку
- разворачивается
- идет в исходное положение
- разворачивается

Можно считать, что в исходное положение робот переходит для получения очередной порции краски.

Такие действия робот выполняет в примере циклически всего 3 раза, что не ограничивает общности.

Программа второго робота скрыта, но она полностью аналогична программе первого робота. Естественно, что через несколько шагов роботы столкнутся, так как попробуют занять одну и ту же клетку. Авария наглядно продемонстрирована (на рис. 8 изображено столкновение роботов из-за отсутствия в программах синхронизации). При этом на рисунке хорошо видна программа для второго робота, которая была скрыта на предыдущем рисунке.

Для решения проблемы взаимоисключения для двух роботов в ПиктоМире потребуется пройти все этапы решения задачи Э.Дейкстры, для начала использовать переменную для синхронизации роботов. Несмотря на простоту ЦОС ПиктоМир в ней имеются простейшие аналоги переменных, в частности кувшин - счетчик. В кувшин можно бросать камушки или доставать их из него, можно проверять состояние кувшина: пуст или не пуст кувшин (у кувшина есть еще несколько команд, которые мы не описали). Фактически кувшин является целочисленным счетчиком, а для решения ряда задач потребуется только значения 0 и 1.

Программа управления роботом содержит все стандартные части: и критический интервал (вперед на выделенную клетку, закрасить, развернуться и сделать шаг) и остаток программы, когда робот возвращается в исходное положение,

разворачивается и снова идет к клетке. Очевидно, что цикл ожидания по условию пока должен выполняться до шага на выделенную клетку. Условие ожидания для робота будет состоять в проверке, пуст кувшин или не пуст.

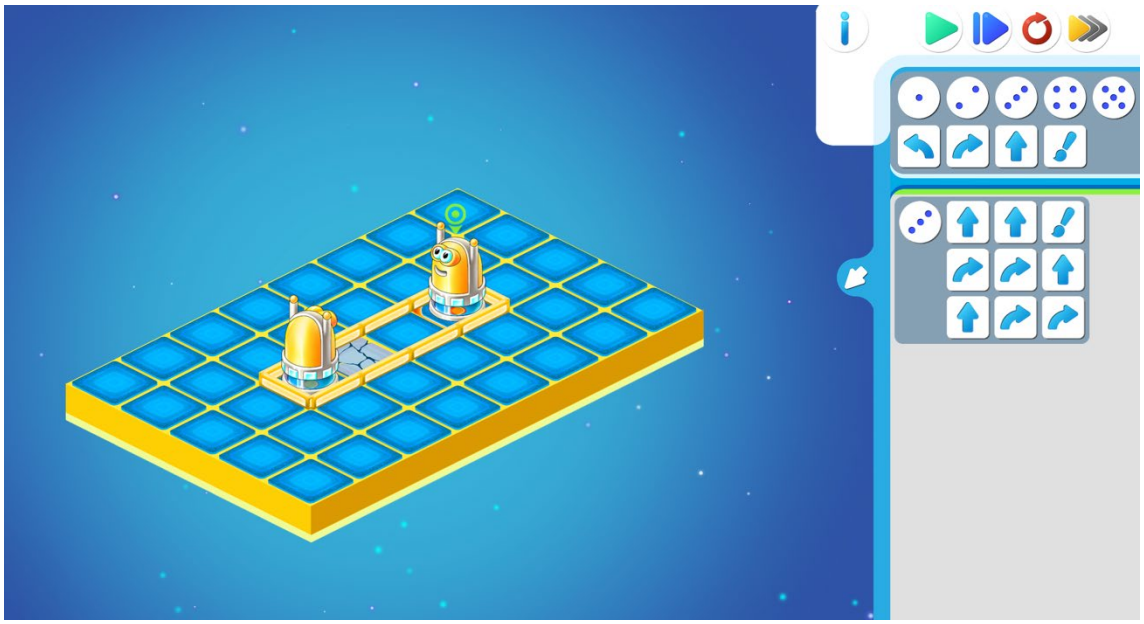

Рис. 7. Исходное состояние и программа для робота, не использующая синхронизацию

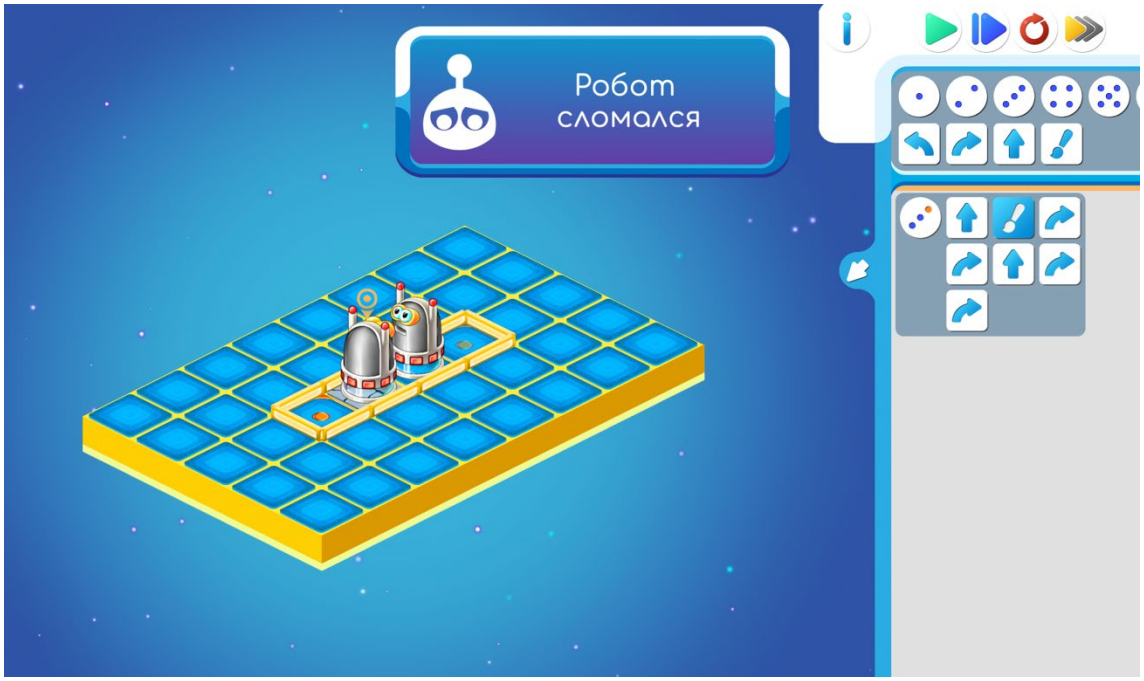

Рис. 8. Роботы столкнулись из-за отсутствия синхронизации

Для ожидания роботы используют дополнительную команду "мигнуть", то есть "пропустить ход", "ничего не делать". Визуально при выполнении команды мигнуть у роботов проскакивает молния между антеннами.

На рис. 9 Роботы красят клетку строго поочередно. Один из роботов занял клетку (находится в своей критической секции), другой выполняет оставшийся код программы, вне критического интервала. Также, как и при реализации алгоритмов синхронизации на Кумире и C++, в программе управления роботом переключается счетчик после покидания критической секции.

Одни робот выбрасывает камень из кувшина, а другой в своей программе кладет камень в кувшин, что соответствует присвоению счетчику 0 или 1. На рис.9 робот, чья программа видна на экране, очищает кувшин.

Рис. 9. Роботы красят клетку строго поочередно.

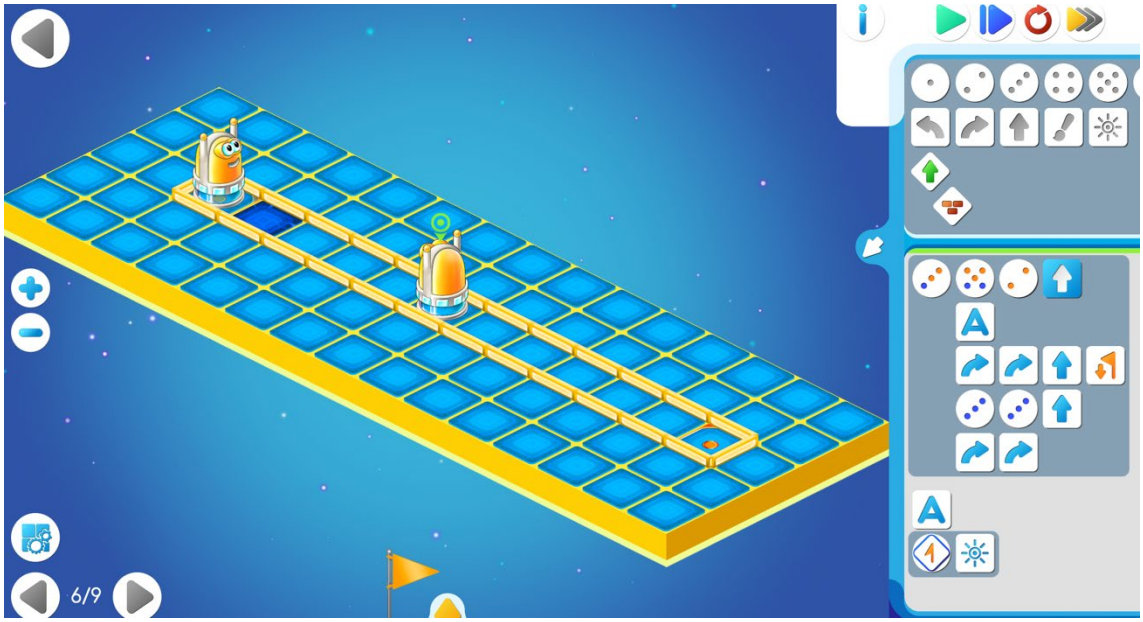

Рис. 10. Решение задачи с флагом.

На рис.10 показана программа поочередного захода на клетку с использованием флага вместо кувшина. Флаг — это битовая (логическая) величина, которая может быть установлена в 0 (флаг опущен) или 1 (флаг поднят). Также можно использовать состояние флага в условии цикла. Обстановка на рисунке демонстрирует низкую эффективность алгоритма синхронизации, поскольку один робот проходит очень длинный путь по сравнению со своим визави. При выполнении программы можно наблюдать робота, находящегося близко от клетки и не имеющего возможности занять совершенно свободную клетку, так как другому роботу требуется пройти только в одну сторону 9 клеток, но составленный алгоритм синхронизации с переключениями предписывает заходить на выделенную клетку строго по очереди: сначала один робот, потом

другой, затем снова один робот, потом снова другой и так далее.

Дальнейшим усовершенствованием алгоритма синхронизации состоит в использовании двух флагов (желтого и зеленого, для черно-белых рисунков левого и правого), поднятие которого сигнализирует о готовности робота занять выделенную клетку.

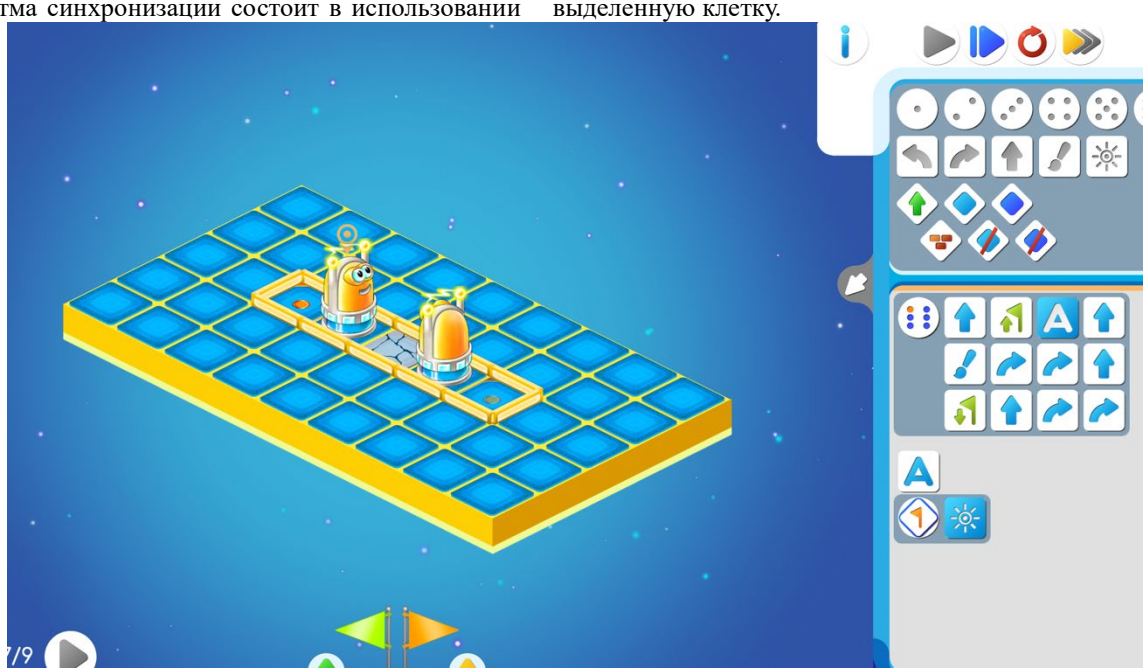

Рис. 11. Роботы находятся в бесконечном ожидании.

Рис. 12. Решение проблемы взаимоисключений.

На рис. 11 роботы находятся в бесконечном ожидании, так как оба были готовы войти в свои критические секции (готовы зайти на клетку), и оба одновременно подняли флаги. При выполнении алгоритма ожидания с проверкой состояния флага (алгоритм А) оба ждут, так как оба флага оказались поднятыми.

На рис. 12 представлено полноценное решение проблемы взаимоисключений, приведен код на языке Пикто, аналогичный соответствующим алгоритмам синхронизации на КуМире и C++. В приведенном решении используется и кувшин (в качестве счетчика, и два флага, назначение которых взято из предыдущего решения. Именно наличие счетчика позволяет избежать проблемы

взаимного блокирования процессов. Для демонстрации решения путь до клетки одного робота в 4 раза превышает путь другого. Таким образом один из роботов чаще приходит в сломанную клетку, чем другой.

Если сравнить размер кода для учебных и производственных языков программирования, то можно с уверенностью сказать, что запись на языке Пикто короче, проще и информативнее.

Рис. 13. Простейшее задание по кооперативному программированию.

5. Заключение

Опыт проведенных занятий со студентами с использованием вышеприведённых алгоритмов синхронизации в ЦОС ПиктоМир для визуализации сложных и важных понятий параллельного программирования, позволяют говорить о повышении эффективности изучения этой важной темы. Студентам впоследствии было легче осваивать методы синхронизации параллельной работы процессов и решать задачи на производственных языках программирования.

Кроме того, картинка результата выполнения задачи в пиктограммной среде программирования ПиктоМир более запоминающаяся и позволяет ученику проще связывать в своем сознании результат работы алгоритма с его содержанием.

Визуализация алгоритмов реализации взаимоисключений ЦОС ПиктоМир стала возможна благодаря внедрений в ПиктоМир режима кооперативно-параллельных заданий. основная идея этого режима состояла в совместном решении поставленных задач двумя и более школьниками, каждый из которых составляет программу для своего робота. Члены команды должны придумать, как согласовать действия своих роботов, договориться с друг другом о порядке выполнения совместной работы.

На рис. 13 представлено простейшее задание для кооперативной работы, когда участникам команды требуется сначала договорится, кто какие клетки красит.

Однако задания по кооперативному программированию можно выполнять и в одиночку, что продемонстрировано изложенным выше решением а ПиктоМире задач на синхронизацию процессов, с использованием графических, материальных аналогов – Кувшин и Флаги – целочисленных и логических переменных.

Задачи на кооперативное программирование в ПиктоМире при решении в одиночку могут восприниматься как головоломки. Такие задания интересны не только школьникам, но и студентам и даже профессиональным программистам. Для этой категории пользователей для синхронизации действий разных роботов в обстановку добавлены исполнитель Кувшин (счетчик) и флаги, доступные во всех потоках параллельной программы. Это позволяет придумывать интересные задачи, над которыми приходится поломать голову даже профессионалам.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0010.

Visualization of Mutual Exclusion Algorithms by Means of a Pictogram Programming Environment

I.N. Gribanova, A.C. Karavaeva, A.A. Leonov, A.G. Leonov, К.А. Мащенко,V.A. Oganisyan

Abstract. Courses in computer science and information and communication technologies can include a wide range of topics from the basics of programming and elementary knowledge about the structure of computers to the theory of formal grammars, artificial intelligence languages, operating systems, 3D graphics, etc. The main difficulty in this case is the mathematical component of the courses, which requires appropriate training of students. However, in programming there is content that is quite difficult for students to understand, even with deep academic knowledge. This area is parallel programming, when, on the one hand, it is required to study and write educational highly efficient algorithms for solving problems for performing simultaneous calculations on different computers or processors, on the other hand, it is no less difficult to understand - ensuring the parallel operation of various parts of a program with a shared access to shared data. The development of the latter is complicated by the lack of adequate visualization of the parallel operation of processes on a computer. It remains for teachers to illustrate the topic with pictures from the real world, giving various arguments to prove the importance of solving the problem of sharing common resources, citing, as one of the options, the simultaneous free moving of four cars at a crossroad. On the other hand, current trends require lowering the age of the beginning of acquaintance with the elements of programming and algorithmization. Preschoolers, children of the fourth year of life, successfully master the basic concepts of sequential programming using program-controlled robots and their virtual counterparts in the digital educational environment PiktoMir. The same DEE PiktoMir is used for programming propaedeutics in teaching for all ages, including university students. PiktoMir is not only easy to learn, but allows the use of several robot performers in tasks at the same time, each of which acts according to its own program. The visibility of the program execution process in the system led the authors to the idea of using the DEE PiktoMir to demonstrate the operation of classical mutual exclusion algorithms in courses on parallel programming. The experience of the authors showed the high efficiency of this approach, which allowed students to understand the problem in a short time and successfully solve all the problems of the course. This approach was tested in the courses Operating Systems (IIS SUM) and Algorithms and Data Structures (Institute of Childhood, Moscow State Pedagogical University).

Keywords: mutual exclusion, process, deadlock, critical section, PiktoMir, operating systems, algorithm.

Литература

Э. Дейкстра. Взаимодействие последовательных процессов // Языки программирования / Ред. Ф. Женуи, пер. с англ. В.П. Кузнецова, под ред. В.М. Курочкина. — М. : Мир, 1972. — С. 9—87.

E. W. Dijkstra. Solution of a problem in concurrent programming control. Communications of the ACM, 8(9):569, 1965.

Taubenfeld, G. Concurrent Programming, Mutual Exclusion. In: Kao, MY. (eds) Encyclopedia of Algorithms. Springer, New York, 2016.

L. Lamport. A new solution of Dijkstra's concurrent programming problem. Communications of the ACM, 17(8):453–455, August 1974.

L. Lamport. The mutual exclusion problem: Part II – statement and solutions. Journal of the ACM, 33:327–348, 1986.

L. Lamport. A bug in the Bakery algorithm. Technical Report CA–7704–0611, Massachusette computer associates, inc., April 1977.

U. Abraham. Bakery algorithms. In Proc. of the Concurrency, Specification and Programming Workshop, pages 7–40, 1993.

Taubenfeld, G. The Black-White Bakery Algorithm and Related Bounded-Space, Adaptive, Local-Spinning and FIFO Algorithms. In: Guerraoui, R. (eds) Distributed Computing. DISC 2004. Lecture Notes in Computer Science, vol 3274. Springer, Berlin, Heidelberg, 2004.

E.W. Dijkstra. Cooperating Sequential Processes in Programming Languages. Ed. F. Genuys. — NY: Academic Press, New York, 1968.

Леонов А.Г., Первин Ю.А., Зайдельман Я.Н. Программные исполнители в цифровых образовательных средах «ПиктоМир», «Роботландия» и «КуМир». Информатика в школе. 2019;(9):54-61.

А. Г. Леонов. Параллельное программирование // Энциклопедия для детей. — Т. 22 из Информатика. — Аванта+ Москва, 2003. — С. 140–145.

Framework Qt. Use Qt's libraries and APIs to develop software with native C++ performance for mobile, desktop, and embedded systems. https://qt.io/ (дата обращения 01.11.2022)

Стартовая страница проекта «ПиктоМир» на сайте ФГУ ФНЦ НИИСИ РАН. URL: https://www.niisi.ru/piktomir/ (дата обращения 01.11.2022)

1. V. B. Betelin, A. G. Kushnirenko, A. G. Leonov, K. A.Mashchenko, Basic Programming Concepts as Explained for Preschoolers, International Journal of Education and Information Technologies (NAUN), Volume 15 (2021): 245-255.

2. N.Besshaposhnikov, A.Kushnirenko, and A.Leonov. Piktomir: how and why do we teach textless programming for preschoolers, first graders and students of pedagogical universities. CEE-SECR '17: Proceedings of the 13th Central & Eastern European Software Engineering Conference in Russia, October 2017. No. 21. P. 1–7. 2017

Комплексная арифметика в ДССП для троичной машины

А.А.Бурцев[1]

[1] ФГУ ФНЦ НИИСИ РАН, Москва, Россия, burtsev@niisi.msk.ru

Аннотация. Созданный в НИЛ троичной информатики на факультете ВМК МГУ программный комплекс ДССП-ТВМ можно использовать в качестве среды разработки и прогона программ для троичного компьютера. Первоначально в нём были предусмотрены лишь операции целочисленной арифметики. Затем ДССП-ТВМ был дополнен пакетом операций вещественной арифметики с плавающей точкой. Теперь на его основе создан пакет комплексной арифметики. В статье характеризуются основные возможности предлагаемого пакета, а также поясняются ключевые аспекты его реализации на языке ДССП-Т в среде интерпретатора ДССП/ТВМ.

Ключевые слова: троичная симметричная система счисления, троичная машина, троичная логика, троичная арифметика, ДССП, вещественная арифметика с плавающей точкой, комплексная арифметика, округление.

1.Введение

В настоящее время, когда двоичная микроэлектроника приблизилась к потолку своих потенциальных возможностей, в качестве одного из перспективных путей дальнейшего развития вычислительной техники исследуется возможность построения цифровых машин на основе троичной симметричной системы счисления (ТССС с цифрами –1,0,+1). Перспективность такого направления была в своё время отмечена ещё Дональдом Кнутом [1], а реальная возможность его осуществления была обозначена практической разработкой и применением троичных цифровых машин «Сетунь» и «Сетунь-70» [2], созданных более полувека назад в НИЛ ЭВМ МГУ под руководством Брусенцова Н.П.

Вопросы построения троичной вычислительной техники по-прежнему являются предметом проводимых научных исследований. Так, в работах Маслова С.П. [3], одного из сотрудников лаборатории Брусенцова Н.П., были предложены способы реализации на современной элементной базе таких аппаратных троичных элементов, которые могли бы функционировать аналогично тем, что были реализованы в машинах семейства «Сетунь». Это вселяет весомую надежду на создание в скором будущем современного троичного компьютера.

В числе достоинств цифровых машин, построенных на основе троичной симметричной системы счисления, обычно отмечают [4-6] следующие её особенности. Троичная – самая экономичная система счисления (подробнее это объясняется, например, в [6]). В ней можно тем же количеством разрядов закодировать больший диапазон чисел и проще реализовать ряд арифметических операций. Можно единообразно обрабатывать как положительные, так и отрицательные числа, и непосредственно реализовать операции троичной логики.

Но для области вычислительной обработки самое ценное качество этой системы счисления проявляется в том, что она позволяет сделать оптимальное округление чисел простым отсечением младших разрядов, а также обеспечивает взаимокомпенсируемость погрешностей округления в процессе вычислений.

Поэтому на троичном компьютере можно выполнять вычисления с лучшей точностью и избавиться, наконец, от назойливых проблем округления, которые были навязаны миру вычислительной техники применением двоичных машин. Такой вывод был сделан специалистами ВЦ МГУ [7] ещё в те годы XX века, когда им предоставлялась возможность решать вычислительные задачи как на двоичных, так и на троичных компьютерах.

В настоящее время в качестве среды разработки и прогона программ для троичного компьютера можно использовать программный комплекс ДССП-ТВМ [8], созданный в НИЛ троичной информатики МГУ. Он включает имитатор троичной машины ТВМ, кросс-компилятор и интерпретатор, позволяя разрабатывать программы для ТВМ на языке ДССП-Т – троичном варианте языка ДССП [9].

Первоначально в ДССП-ТВМ можно было создавать троичные программы, используя лишь операции целочисленной арифметики. Затем ДССП-ТВМ была дополнена пакетом операций вещественной арифметики с плавающей точкой [10]. А на его основе теперь создан пакет комплексной арифметики.

В итоге в настоящее время в ДССП-ТВМ

обеспечена возможность разрабатывать для троичной машины и прогонять на ней вычислительные программы, в которых требуется обрабатывать как вещественные, так и комплексные числа.

В статье характеризуются основные возможности предлагаемого пакета комплексной арифметики, а также поясняются важнейшие аспекты его реализации на языке ДССП-Т в среде интерпретатора ДССП/ТВМ.

2. Общая характеристика пакета комплексной арифметики

Данный пакет комплексной арифметики предлагается использовать в среде интепретатора ДССП-ТВМ, который уже оснащён операциями вещественной арифметики. После загрузки этого пакета командой **LOAD Cmplx** (из файла **Cmplx.dsp**) будет подготовлена новая среда интерпретатора с операциями комплексной арифметики, версию которой затем можно сохранить командой **SAVE** (в файле **DSSP-3dC.nth**) :

```
****** DSSP/TVM *******  v.3d
* LOAD Cmplx
* SAVE DSSP-3dC
```

Подлежащие обработке значения комплексных чисел занимают в стеке две позиции: одна из них представляет вещественное значение реальной части комплексного числа, а другая – мнимой. А каждое из этих вещественных значений, в свою очередь, представляется в 27-тритном машинном слове как вещественное число в форме с плавающей точкой. При этом значение порядка помещается в его старшем трайте (в разрядах с 18-го по 26-й), а значение мантиссы располагается в двух младших трайтах (в разрядах с 0-го по 17-й) такого троичного слова (подробнее об этом см. в п.3.1 в [10]).

Все предусмотренные в пакете операции с комплексными числами действуют над величинами, помещёнными в стек. При загрузке комплексного значения в стек его реальная часть помещается в подвершину, а мнимая часть – в вершину стека.

Поэтому одноместные операции комплексной арифметики затрагивают две верхние позиции стека, двухместные операции – четыре, а операции, требующие 3 аргумента, действуют соответственно над шестью верхними позициями стека.

Чтобы было удобно копировать и переставлять в стеке комплексные числа, в пакете предусмотрены соответствующие дополнительные операции, позволяющие обрабатывать позиции стека парами. Представляя верхние позиции стека так, как будто они содержат одно, два или три комплексных числа, операции **CC CC2 CC3** позволяют скопировать наверх стека комплексное значение, лежащее в нём на глубине соответственно 1, 2 или 3:

```
{Re,Im} CC { Re,Im, Re,Im }
{Re,Im,a,b} CC2 {Re,Im,a,b,Re,Im}
{Re,Im,a,b,c,d} CC3 {Re,Im,a,b,c,d,Re,Im}
```

А операции **CE2** и **CE3** позволяют переставить в стеке комплексные числа, лежащие на глубине 1-2 и 1-3 соответственно:

```
{XRe,XIm,YRe,YIm } CE2 {YRe,YIm,XRe,XIm}
{XRe,XIm,a,b,YRe,YIm} CE3
                      {YRe,YIm,a,b,XRe,XIm}
```

Комплексное значение состоит из пары вещественных и занимает в стеке две 27-тритных позиции. И для занесения такого парного значения в стек из памяти по указанному адресу (A), а также для записи такой пары из стека в память тоже предусмотрены аналогичные дополнительные операции **@C !C** :

```
{A} @C {Re,Im}   { Re=M[A], Im=M[A+3] }
{Re,Im,A} !C { } { M[A]=Re, M[A+3]=Im }
```

В пакете представлены две одноместные операции над комплексным числом. Операция, обозначаемая словом **CNEG**, изменяет знак числа, а операция **C'** получает из него сопряжённое комплексное значение:

```
{XRe,XIm} CNEG {–XRe,–XIm}
{XRe,XIm} C' {XRe,–XIm}
```

Для сложения и вычитания двух комплексных чисел предусмотрены двухместные операции, обозначаемые словами **C+** и **C–** :

```
{XRe,XIm,YRe,YIm} C+ {XRe+YRe,XIm+YIm}
{XRe,XIm,YRe,YIm} C- {XRe–YRe,XIm–YIm}
```

Операция умножения двух комплексных чисел осуществляется словом **C*** :

```
{XRe,XIm,YRe,YIm} C*
{XRe*YRe–XIm*YIm, XRe*YIm+XIm*YRe}
```

Предусмотрены также операции, сочетающие умножение двух комплексных чисел вместе со сложением (или вычитанием) получаемого результата с третьим заданным числом. Они обозначаются словами **C*+** и **C*–** :

```
{ZRe,ZIm, XRe,XIm,YRe,YIm} C*+
{ ZRe + (XRe*YRe–XIm*YIm),
  ZIm + (XRe*YIm+XIm*YRe) }
{ZRe,ZIm, XRe,XIm,YRe,YIm } C*-
{ ZRe – (XRe*YRe–XIm*YIm) ,
  ZIm – (XRe*YIm+XIm*YRe) }
```

Заметим, что эти операции используют три помещённых в стек комплексных значения, оставляя вместо них в стеке лишь одно комплексное значение в качестве результата.

В пакет включена и самая нагруженная операция комплексной арифметики, совмещающая в себе умножение со сложением и вычитанием

комплексных чисел, которая одна реализует сразу обе рассмотренные выше операции. Она обозначается словом **C*+-** :

```
{ZRe,ZIm, XRe,XIm,YRe,YIm}  C*+-
{ ZRe + (XRe*YRe–XIm*YIm) ,
  ZIm + (XRe*YIm+XIm*YRe) , }
{ ZRe – (XRe*YRe–XIm*YIm) ,
  ZIm – (XRe*YIm+XIm*YRe) }
```

Эта операция тоже использует три помещённых в стек комплексных значения, но оставляет вместо них в стеке не одно, а два комплексных значения в качестве результата.

Заметим, что вызовом слова **C*+-** можно исполнить одну из важнейших базовых операций, применяемых в задачах обработки сигналов, которая получила образное название «бабочка Фурье».

В пакете имеются также операции, в которых наряду с комплексными числами в качестве параметров участвуют и вещественные числа. Так, для умножения обеих компонентов комплексного числа на вещественное значение предназначена операция **C*S** :

```
{ XRe,XIm,S }  C*S { XRe*S,XIm*S }
```

Операция, обозначаемая словом **CMOD^2**, позволяет получить вещественное значение, являющееся квадратом модуля заданного комплексного числа:

```
{ XRe,XIm }  CMOD^2 { XRe^2+XIm^2 }
```

А операция, обозначаемая словом **|C|**, позволяет получить сумму абсолютных значений его реальной и мнимой частей:

```
{ XRe,XIm }  |C| { |XRe|+|XIm| }
```

Для обеспечения ввода и вывода комплексных чисел в привычном десятичном виде предусмотрены операция **Str->Cmplx** для преобразования строки символов в комплексное значение (состоящее из двух вещественных), а также ряд операций для печати комплексного значения: **.C**, **.Cmplx** (с удалением его из стека) и **.Cmplxp** (с указанием числа позиций):

```
{Str,Len} Str->Cmplx {Re,Im}
{Re,Im} .C  {Re,Im}
{Re,Im} .Cmplx  { }
{Re,Im,p} .Cmplxp  { }
```

При этом строка, изображающая комплексное число, представляется в привычном формате (см. табл. 1) в виде двух вещественных чисел, соединённых знаком + или –, второе из которых завершается символом мнимой единицы (**i** или **I**). Вещественное число реальной части может отсутствовать (тогда оно полагается равным нулю 0.). А при отсутствии числа в мнимой части (перед символом мнимой единицы), оно полагается равным единице (1.0).

В случае невозможности преобразования заданной строки в комплексное число (при исполнении операции **Str->Cmplx**), возбуждается исключительная ситуация **WrongCmplx!**.

Таблица 1.Форматы комплексного числа

Форматы	Примеры
±R±Ri	3.–2i 1E–2+1.i –5.4+3.2I 1.2+1E+3i
±Ri	+3.i –1E–3i
±R±i	1.+i –2E+3–I
±i	+I –I +i –i I i
± - возможный знак + или – **R** – вещественное число (без знака) **i** – обозначение мнимой единицы: i или I	

Для некоторых особо употребительных комплексных констант в словарь ДССП пакетом уже занесены представляющие их слова:

```
i  {0.,1.0}     -i  {0.,-1.0}
1+0i  {1.,0.}  -1+0i  {-1.,0.}
```

Для обеспечения возможности определения именованных комплексных констант создано компилирующее слово **DVALUE**, позволяющее задать слово-константу, при вызове которого в стек посылается два представляющих его 27-тритных значения.

Например, в результате такого определения:

```
1. -1. DVALUE 1-i
```

в словарь ДССП добавляется слово **1-i**, вызов которого:

```
1-i {+1.0,–1.0}
```

заносит в стек два вещественные значения 1.0 (в подвершину) и –1.0 (в вершину), представляющие комплексное значение 1.0–1.0i :

В пакете также определяется новый тип данных **COMPLEX**, что позволяет объявлять в ДССП-программе переменные и массивы для работы с комплексными числами.

Представим примеры подобных объявлений:

```
COMPLEX VAR X   COMPLEX VAR Y
32 COMPLEX VCTR V  { V[0:32] }
```

И продемонстрируем типовые действия с переменными и векторами комплексного типа:

```
X  {XRe,XIm}  ! Y  {Y:=X}
1 V {V[1].Re,V[1].Im}  31 ! V { V[31]:=V[1]}
```

В первом примере вызов имени переменной **X** типа **COMPLEX** засылает в стек два вещественных значения XRe и XIm, представляющих реальную и мнимую часть её комплексного значения. А командой, обозначаемой словом **!**, эти два вещественных значения помещаются в реальную и мнимую часть переменной **Y**. А во втором примере комплексное значение, взятое из элемента вектора **V**[1], записывается как новое значение для элемента **V**[31].

3. Описание реализации пакета комплексной арифметики

Рассмотрим поподробнее ключевые аспекты реализации представленного пакета комплексной арифметики.

3.1. Реализация основных операций комплексной арифметики

В настоящее время все операции над троичными комплексными числами реализованы в ДССП-ТВМ на основе операций вещественной арифметики на языке ДССП-Т. Продемонстрируем приёмы реализации основных операций комплексной арифметики.

Реализация одноместной операции **C'** получения сопряжённого комплексного значения заключается в смене знака вещественного значения лишь его мнимой части, а для реализации операции смены знака **CNEG** требуется поменять знак у обеих вещественных компонент комплексного значения:

```
: C' {Re,Im} RNEG {Re,–Im} ;
: CNEG {Re,Im} RNEG {Re,–Im}
        {Re,–Im} E2 RNEG E2 {–Re,–Im} ;
```

Реализация одноместных операций **CMOD^2** и **|C|** сводится к вычислению одного вещественного значения с применением операций **R+ RABS** и **R*** над вещественными значениями его реальной и мнимой частей:

```
: CMOD^2 {Re,Im} C R*
          E2 C R* R+ {Re^2+Im^2} ;
: |C| {Re,Im} RABS
          E2 RABS R+ {|XRe|+|XIm|} ;
```

Реализация двухместных операций сложения **C+** и вычитания **C–** заключается в исполнении вещественных операций сложения **R+** и вычитания **R–** по отдельности над их вещественными компонентами:

```
: C+ {XRe,XIm,YRe,YIm} E4 R+
       E3 R+ {XRe+YRe,XIm+YIm} ;
: C- {XRe,XIm,YRe,YIm} E4 E2 R-
       E3 R- {XRe–YRe,XIm–YIm} ;
```

Умножение комплексного значения на вещественное значение осуществляется применением вещественной операции **R*** для умножения на это значение каждой из его вещественных компонент:

```
: C*S {Re,Im,S} E3 C3 R* E3
      {Re*S,Im,S} R* {Re*S,Im*S} ;
```

А вот при реализации операции комплексного умножения приходится уже воспользоваться всеми тремя вещественными операциями умножения **R***, сложения **R+** и вычитания **R–**, чтобы получить итоговые вещественные значения реальной и мнимой частей результата комплексного умножения по известным формулам:

```
: C* {XRe,XIm,YRe,YIm} C4 C3 R*
    C4 C3 R* R- {.., XRe*YRe–XIm*YIm }
    5 ET R* E3 R* R+
 {XRe*YRe–XIm*YIm, XRe*YIm+XIm*YRe} ;
```

На основе этих базовых операций комплексной арифметики (уже реализованных **C+ C– C***) можно далее реализовать более сложные комплексные операции умножения с накоплением:

```
{ Z=(Z.Re,Z.Im),X=(X.Re,X.Im),Y=(Y.Re,Y.Im) }
: C*+ {Z,X,Y} C* C+ {Z'=Z+X*Y} ;
: C*- {Z,X,Y} C* C- {Z'=Z–X*Y} ;
: C*+- {Z,X,Y} C* CE2 {X*Y,Z}
    CC2 CC2 C+ CE3 C- {Z+X*Y,Z–X*Y} ;
```

Заметим, что здесь используются также операции копирования и перестановки в стеке комплексных значений, занимающих в стеке пару позиций. Продемонстрируем приёмы реализации таких операций над парами позиций стека:

```
: CC2 {Re,Im,a,b} C4 C4 {Re,Im,a,b,Re,Im} ;
: CE2 {XRe,XIm,YRe,YIm} E3
      E2 E4 E2 {YRe,YIm,XRe,XIm} ;
: CE3 {XRe,XIm,a,b,YRe,YIm} E2 6 ET
      E2 5 ET {YRe,YIm,a,b,XRe,XIm} ;
```

А также представим реализацию операций чтения комплексных значений из памяти в стек и записи из стека в память:

```
: @C {Adr} C @W E2 3+ @W {Re,Im} ;
        { Re=Mem[Adr], Im=Mem[Adr+3] }
: !C {Re,Im,Adr} E2 C2 3+ !W !W { } ;
        { Mem[Adr]=Re, Mem[Adr+3]=Im }
```

3.2. Представление комплексных значений в виде символьных строк

Чтобы полученные в результате вычислений комплексные значения представить в визуальном виде, необходимо уметь выводить их на печать (например, на экран терминала). Операции печати комплексного значения используют уже имеющиеся операции печати вещественного значения и дополняют образуемую ими строку вывода до изображения комплексного значения в виде двух соединённых символом знака вещественных значений, оканчивающихся символом мнимой единицы:

```
: .Cmplxp {Re,Im,p}
   E3 C Rm=0? BR+ D .Re.Realp C
   |Im|=1? BR+ .Im* .Im.Realp .'i' { } ;
 : .Re.Realp {p,Im,Re} C3 .Realp {p,Im} ;
 : |Im|=1? {Im} RABS 1.0 R= {|Im|=1.0?} ;
 : .Im.Realp {p,Im} C RSGN
    BRS NOP .'+' .'+' E2 .Realp { } ;
 : .Im* {p,Im} RSGN
    BRS .'-' .'+0' .'+' D { } ;
  : .'-' '-' TOB ;  : .'+' '+' TOB ;
  : .'i' 'i' TOB ;
  : .'+0' '+' TOB '0' TOB ;
: .Cmplx {Re,Im} 8 .Cmplxp { } ;
```

```
: .C {Re,Im} CC .Cmplx {Re,Im} ;
```

Для ввода комплексного числа необходимо уметь преобразовывать вводимую строку в комплексное значение. При реализации такой операции преобразования **Str->Cmplx** два раза вызывается операция преобразования строки в вещественное значение **Str->Real** (сначала для реальной, а потом для мнимой части), а также осуществляется проверка соблюдения формата для задаваемого комплексного числа (согласно таблице 1):

```
VAR pBCS { указатель на начало заданной строки }
VAR pECS { указатель на конец заданной строки }
VAR pCS { указатель на текущий символ строки }
: Str->Cmplx {Str,Len}
   EON WrongCmplx! RERAISE
   C2 ! pBCS C2 C2 + ! pECS
   {Str,Len} Get.Re 0. {XRe,XIm=0.}
   Str->Real_pS ! pCS Str->Real_Ch {Ch}
   BR '+' Get.ImReal '-' Get.ImReal
      'i' E2 'I' E2 'SP' NOP NUL NOP
   ELSE WrongCmplx! {XRe,XIm)} ;
: Get.Re {ReStr,Len} Str->Real {XRe} ;
: Get.ImReal {XRe,0.} pCS 1- pECS C2 -
   Get.Im E2D Str->Real_pS ! pCS
   Str->Real_Ch =I? IF0 WrongCmplx!
   Ch=SP|NUL? IF0 WrongCmplx! {XRe,XIm} ;
 : Get.Im {ImStr,ImLen} Str->Real {XIm} ;
 : =I? {Ch} C 'i' = E2 'I' = OR ;
 : Ch=SP|NUL? pCS pECS C2 <=
    {pCS,pECS<=pCS?} BR+ T1 =SP? ;
   : =SP? @T 'SP' = ;
```

Вместе с вызовами процедуры **Str->Real** используются также вспомогательные процедуры Str->Real_pS и Str->Real_pS, которые позволяют узнать, где в заданной строке и на каком символе остановилось распознавание вещественного числа с тем, чтобы иметь возможность продолжать в заданной строке дальнейшее распознавание комплексного значения.

3.3. Реализация типа COMPLEX

Реализация типа данных, представляющего комплексное значение, осуществлялась с применением имеющихся в ДССП-ТВМ средств объектно-ориентированного программирования, подробно описанных в [11].

Тип данных **COMPLEX**, используемый для объявления переменных и массивов комплексного типа, определяется как класс с полями, представляющими вещественные значения его реальной и мнимой частей, и методами, предназначенными для чтения комплексного значения в стек и записи нового значения из стека:

```
CLASS: COMPLEX
          VAR .Re  VAR .Im
        0 METHOD# @Val
        1 METHOD# !Val
;CLASS
```

Процедура, вызываемая по имени объекта (типа **COMPLEX**), как исполнитель метода с номером 0, в начале своего выполнения получает в вершине стека адрес объекта, используя который, считывает два вещественных значения из полей объекта **.Re** и **.Im**, чтобы занести их соответственно в подвершину и вершину стека:

```
COMPLEX :M: @Val
  {'X} C .Re E2 .Im {X.Re,X.Im} ;
```

А процедура, исполняющая метод (с номером 1), вызываемый префиксной операцией над объектом, обозначаемой словом **!** , приготовленную в стеке пару вещественных значений Re и Im записывает в память (соответственно в поля **.Re** и **.Im**) по адресу объекта, передаваемого ей в вершине стека:

```
COMPLEX :M: !Val { X.Re:=Re; X.Im:=Im }
  {Re,Im,'X} E2 C2 ! .Im ! .Re { } ;
```

3.4. Реализация компилирующего слова DVALUE

В ассортименте компилирующих средств ДССП-ТВМ уже предусмотрено слово **VALUE**, позволяющее создавать именованные слова-константы, при вызове которых в стек засылается одно 27-тритное значение, определённое при объявлении такой константы.

Но комплексная константа состоит из двух вещественных значений. Значит, при вызове слова, представляющего именованную комплексную константу, в стек должно засылаться два вещественных 27-тритных значения.

Поэтому возникла необходимость создать в ДССП-ТВМ новое компилирующее слово (названное **DVALUE** по аналогии со словом **VALUE**), позволяющее определять слова-константы, которые при вызове имени посылали бы в стек не одно, а два 27-тритных значения.

Приведём реализацию этого компилирующего слова в среде интерпретатора ДССП-ТВМ на языке ДССП-Т:

```
: DVALUE HEAD {Val1,Val2} ''LIT , E2 ,
  {Val2} ''LIT , {Val2} , ''; , { } ;
```

Далее при вызове такого слова **DVALUE**:

```
Val1 Val2 DVALUE DCNST
```

будет создаваться слово DCNST, посылающее в стек два значения Val1 и Val2, тело которого будет сформировано, как и при определении:

```
: DCNST Val1 Val2 ;
```

Реализация подобных компилирующих слов в среде интерпретатора осуществляется при помощи комплекта вспомогательных компилирующих средств, предусмотренных в ДССП-ТВМ (и описанных в [12, п.7.5.]). А с приёмами построения и функционирования тел, формируемых такими словами-компиляторами, можно подробно ознакомиться в [13].

4. Примеры применения пакета комплексной арифметики

Проиллюстрируем применение операций комплексной арифметики примерами разработки в ДССП-ТВМ программ, выполняющих дискретное преобразование Фурье над векторами комплексных чисел длиной $N=2^P$.

Сначала приведём реализацию процедуры простого дискретного преобразования Фурье (ДПФ) квадратичной сложности $O(N^2)$, которая вычисляет комплексные значения результирующего вектора напрямую по приведённым далее формулам. А затем рассмотрим процедуру быстрого преобразования Фурье (БПФ) логарифмической сложности $O(N \cdot \log_2 N)$, исполнение которой осуществляется путём многократного применения операции комплексной арифметики, называемой «бабочка Фурье».

4.1. Пример реализации процедуры простого преобразования Фурье

Операция дискретного преобразования Фурье над заданным вектором комплексных чисел Y_N заключается в получении результирующего вектора X_N комплексных чисел, элементы которого X_k вычисляются на основе исходного вектора Y по правилу:

$$X_k = \sum_{j=0}^{N-1} \{Y_j \times W_N^{k \cdot j}\}$$

в котором так называемые весовые коэффициенты вида W^q_N вычисляются по формуле: $W_N^q = e^{-i \cdot 2\pi \cdot q/N}$ или $W_N^q = \cos\frac{2\pi q}{N} - i \cdot \sin\frac{2\pi q}{N}$, т.к. $e^{i\cdot\varphi} = \cos\varphi + i \cdot \sin\varphi$, где i – мнимая комплексная единица.

Будем считать, что эти величины вычислены заранее и приготовлены в виде массива комплексных чисел W размером от 0 до N. Для осуществления ДПФ можно предложить простой алгоритм, состоящий из двух вложенных циклов, который вычисляет каждый элемент результирующего вектора напрямую по описанному правилу. Выразим этот алгоритм отдельной функцией **DFT** на языке С:

```
void DFT(int N, complex *X,
       complex *Y, complex *W) {
int k,j; complex Xk, V;
for (k=0; k<N; k++) {
 Xk= 0.0 + I*0.0;
 for (j=0; j<N; j++) {
   V=W[(k*j)%N]; Xk= Xk+Y[j]*V;
 }//for j
 X[k]= Xk;
}//for k
}//DFT
```

А теперь продемонстрируем реализацию по такому же алгоритму процедуры **DFT** в среде ДССП-ТВМ с применением пакета комплексной арифметики:

```
: DFT {W,Y,X,N} C DO- X[k] DDDD { } ;
: X[k] {W,Y,X,N,k'} {k'=N-1,..,1,0}
   C2 1- C2 - 0. 0. {..,k',k=N-1-k',Xk=(0.,0.)}
   5 CT DO- +X[k] {..,k',k,Xk=(Re,Im)}
   C3 6 * 7 CT +  {..,'X[k]=X+6k}
  {W,Y,X,N,k',k,Xk,'X[k]} !C {X[k]:=Xk}
   D {W,Y,X,N,k'} ;
: +X[k] {W,Y,X,N,k',k,Xk=(Re,Im),j'} {j'=N-1,..,1,0}
   6 CT 1- C2 - {..,j=N-1-j'}
   W[(k*j)%N] {..,k',k,(Xk),j',j,V=(VRe,VIm)}
   C3 6 * 12 CT + {..,'Y[j]} @C
  {..,(Xk),j',j,(V),(Y[j])} C* CC3 C+ CE3 DDD
  {W,Y,X,N,k',k,(Xk)=(Xk)+(V)*(Y[j]),j'} ;
: W[(k*j)%N] {W,Y,X,N,k',k,Xk=(Re,Im),j',j}
   C 6 CT * 8 CT /DivMod E2D {..,(k*j)%N}
   6 * 11 CT + @C {..,V=W[(k*j)%N] } ;
```

В этом алгоритме для получения результирующего вектора длины N приходится исполнять порядка $O(N^2)$ операций с комплексными числами. Вот почему такой алгоритм характеризуется вычислительной сложностью $O(N^2)$.

4.2. Пример реализации процедуры быстрого преобразования Фурье

Для выполнения БПФ будем использовать вариант известного алгоритма Кули-Тьюки с прореживанием по времени для векторов комплексных чисел длиной $N=2^P$. В этом алгоритме сначала выполняется так называемое бит-реверсное копирование (или перестановка) вектора, а затем в несколько ($P=\log_2 N$) этапов осуществляется операция «бабочка Фурье» (**BF**) над всевозможными парами элементов вектора.

Подробнее с таким алгоритмом можно ознакомиться, например, в п.2 в [14]. Здесь же мы ограничимся его описанием в виде отдельной функции на языке Си:

```
void FFT(int N, complex *X,
 complex *Y,complex *W)
{long int P,t,s,h,G,R,d,k,u,j,a,b;
 complex V,T;
// Часть 1. Бит-реверсное копирование Y=>X:
 BRVPRST(N,X,Y);
//ч2.Основной цикл исполнения бабочек Фурье:
 P=iLog2(N); // P= log2N (так что N=2^P)
 s=1;h=2;G=1;R=N/2;d=N/2;
 for(t=1;t<=P;t++) { // на t-ом этапе:
  for(k=0,u=0;k<G;k++,u=u+d){
   V=W[u];
   for(j=0,a=k;j<R;j++,a=a+h)
   {b=a+s; BF(X[a],X[b],V,T);}
  }//for k
  h=h*2;s=s*2;G=G*2;R=R/2;d=d/2;
 }//for t
}//FFT
```

Для заданного комплексного вектора **Y** длиной **N** и таблицы комплексных коэффициентов

W функция **FFT** получает в качестве результата комплексный вектор **X**. При своём исполнении она вызывает процедуру бит-реверсного копирования **BRVPRST** (которая здесь не уточняется) и многократно осуществляет над парами комплексных элементов операцию **BF** («бабочку Фурье»), которую можно выразить в виде такого макроопределения на языке Си:

```
#define BF(XA,XB,V,Tmp) \
 { Tmp=XB*V; XB=XA-Tmp; XA=XA+Tmp; }
```

На каждом из $P=\log_2 N$ этапов алгоритма БПФ операция **BF** осуществляется над всеми N/2 парами элементов вектора, поэтому вычислительная сложность такого алгоритма оценивается величиной $O(N\log_2 N)$.

А теперь продемонстрируем реализацию процедуры **FFT**, действующей по такому алгоритму БПФ, в среде ДССП-ТВМ с применением пакета комплексной арифметики:

```
: FFT {W,Y,X,N} {{ P= log2(N), так что N=2^P
 {{ч1. Бит-реверсное копирование X<=BRVCopy(Y)
   'BRT C4 C4 C4 BrvCopyCVctr
 {{ч2. Основной цикл исполнения бабочек Фурье:
   1 2 1 N /2 C {..,s=1,h=2,G=1,R=N/2,d=N/2}
   P DO- DoStage(t) {t'=P-1,..,1,0; t=0,1,..P-1}
   DDD DD DDDD { } ;
: DoStage(t) {W,Y,X,N,s,h,G,R,d,t'}
   0 5 CT {..,u=0,G} DO- DoGroup(k) D
   5 ET 6 ET 4 * 5 ET
   E4 *2 E4 E3 /2 E3 E2 /2 E2
 {W,Y,X,N, s=s*2,h=2*s,G=G*2,R=R/2,d=d/2,t'} ;
: DoGroup(k) {W,Y,X,N, s,h,G,R,d,t',u,k'}
   C2 6 * 13 CT + @C {..,(V)=W[u]}
   8 CT 1- C4 - {..,(V),a=k}
   8 CT {R} DO- DoBTFLY(j) DDD
   {..,d,t,u,k'} E2 C4 + E2 {..,d,t,u=u+d,k'} ;
: DoBTFLY(j) {W,Y,X,N, s,h,G,R,d,t',u,k',(V),a,j'}
   12 CT C3 + {..,b=a+s} 15 CT C C3 6 * +
   E3 D C4 6 * + {..,(V),a,j','X[b],'X[a]}
   C @C C4 @C 10 CT 10 CT
  {..,(V),a,j','X[b],'X[a],Xa=(X[a]),Xb=(X[b]),(V)}
   BF {..,(V),a,j','X[b],'X[a], (Xa$),(Xb$)}
   6 CT !C {X[b]:=(Xb$)} C3 !C {X[a]:=(Xa$)}
  {..,(V),a,j','X[b],'X[a]} DD E2 11 CT + E2
  {..,(V),a'=a+h,j'} ;
: BF {(Xa),(Xb),(V)} C*+- {(newXa),(newXb)} ;
```

4.3. Сравнение результатов разных алгоритмов преобразования Фурье

В отличие от целых чисел, вычисления над которыми в компьютере можно осуществлять точно (в рамках разрешённого диапазона), с вещественными числами (представленными в форме с плавающей точкой) произвести точных вычислений на компьютере, вообще говоря, не удаётся. Можно рассчитывать получить лишь такой результат, который с той или иной степенью погрешности будет совпадать с действительным.

Другими словами, все вычисления, совершаемые с такими вещественными числами в компьютере, осуществляются не точно, а приближённо. Почему так? Во-первых, само значение вещественного числа в форме с плавающей точкой уже представляется в компьютере с некоторой погрешностью. А во-вторых, многие операции над такими числами осуществляются с применением округления.

К сожалению, из-за накапливаемых погрешностей округления возможны даже расхождения в результатах вычислений, которые выполняются по одинаковым формулам, но разными способами (алгоритмами), предполагающими различный порядок исполнения операций (например, другой порядок суммирования слагаемых).

Подобное расхождение вычислительных результатов наблюдается и при выполнении расчётов с применением комплексных чисел (представляемых парой вещественных). Например, вектора комплексных чисел, получаемые в результате преобразования Фурье двумя разными алгоритмами (ДПФ и БПФ), могут в точности и не совпадать.

Поэтому сравнивать результаты вычислений, полученные разными способами, можно лишь с некоторой точностью, оговаривая при этом определённую степень близости значений (вещественных или комплексных) чисел, при соблюдении которой считать их равными.

Для вещественных чисел в качестве меры близости обычно принимают абсолютную величину разности их значений. И полагают, что два числа X и Y совпадают с заданной точностью EPS, если соблюдается условие: $|X-Y| \le EPS$.

В качестве меры близости двух комплексных значений $X=(X.Re,X.Im)$ и $Y=(Y.Re,Y.Im)$ обычно принимают модуль их разницы, вычисляемый как расстояние между ними на комплексной плоскости по формуле:

$$M(X,Y) = \sqrt{(X.Re - Y.Re)^2 + (X.Im - Y.Im)^2}$$

И полагают два комплексных числа X и Y равными с точностью EPS при соблюдении условия: $M(X,Y) \le EPS$.

С использованием такого критерия совпадения комплексных значений были проведены сравнения комплексных векторов, полученных в качестве результатов преобразования Фурье рассмотренными выше алгоритмами ДПФ и БПФ. Прогоны и сравнения результатов выполнялись на двоичной машине архитектуры Intel и на троичной виртуальной машине ТВМ.

Вычисления на двоичной машине осуществлялись программой на языке Си, подготовленной в среде MS Visual Studio 2017, в которой для представления комплексных чисел (типа complex) использовалась пара вещественных 32-битных значений одинарной точности (типа float).

Для исполнения ДПФ и БПФ в этой программе вызывались Си-функции DFT и FFT, представленные в п.4.1. и 4.2.

Вычисления на троичной машине выполнялись в среде интерпретатора ДССП-ТВМ с использованием операций пакета комплексной арифметики, в которых комплексные числа представлены двумя вещественными 27-тритными значениями. Для исполнения ДПФ и БПФ в ДССП-программе вызывались слова-процедуры DFT и FFT, реализация которых была представлена ранее в п.4.1. и 4.2.

Для выявления несовпадений комплексных значений в ДССП-программе определялось слово **CmpCVError?** :

```
: CmpCVError? {XRe,XIm,YRe,YIm}
   CC2 CC2 C- CMOD^2 CEPS C R* R>
   {((XRe−YRe)^2+(XIm−YIm)^2)>CEPS^2};
```

которое для двух комплексных чисел X=(XRe,XIm) и Y=(YRe,YIm), размещённых в стеке, проверяло, совпадают ли они с заданной точностью CEPS. А проверка на несовпадение выполнялась по упрощённой формуле:

$$((XRe - YRe)^2 + (XIm - YIm)^2) > CEPS^2$$

без вычисления квадратного корня.

И в Си-программе (на двоичной машине) и в ДССП-программе (на троичной машине) преобразование Фурье выполнялось над векторами комплексных чисел длиной вида $N=2^P$ для значений N от 32 до 4096, а исходный вектор Y предварительно инициализировался по одному и тому же правилу:

Y[k].Re= +**cos**(−π+k·h),

Y[k].Im= −**sin**(−π +k·h),

где h= 2·π/N , k=0,1,..,N-1

Комплексные вектора, полученные в результате выполнения преобразования Фурье алгоритмами DFT и FFT, сравнивались поэлементно с точностью EPS= 10^{-4}, 10^{-5}, 10^{-6}. При этом для каждой пары таких векторов подсчитывалось количество элементов, в которых комплексные значения не совпадали с заданной точностью.

Выявленное в итоге проведённых сравнений количество несовпадений, характеризующее расхождение результатов ДПФ и БПФ для векторов длины вида $N=2^P$ на двоичной машине (Intel) и троичной машине (ТВМ), представлены соответственно в таблицах 2 и 3.

Таблица 2. Расхождение результатов ДПФ и БПФ на двоичной машине архитектуры Intel

длина		точность сравнения **EPS**		
P	**$N=2^P$**	**10^{-4}**	**10^{-5}**	**10^{-6}**
5	32	0	0	1
6	64	0	0	11
7	128	0	1	47
8	256	0	4	121
9	512	0	18	329
10	1024	4	51	783
11	2048	9	188	1759
12	4096	32	551	3802

Таблица 3. Расхождение результатов ДПФ и БПФ на троичной виртуальной машине ТВМ

длина		точность сравнения **EPS**		
P	**$N=2^P$**	**10^{-4}**	**10^{-5}**	**10^{-6}**
5	32	0	0	2
6	64	0	0	4
7	128	0	1	18
8	256	0	0	32
9	512	0	3	106
10	1024	1	16	246
11	2048	1	49	713
12	4096	3	110	2040

Анализируя данные, представленные в этих таблицах, можно сделать следующие выводы.

1. При установлении более строгого критерия равенства комплексных значений, выражающегося в задании меньшего значения EPS, возрастает количество выявляемых случаев несовпадения значений в векторах-результатах.

2. С ростом длины векторов (N) возрастает и количество обнаруживаемых ошибок несовпадения значений в векторах-результатах, что объясняется как следствие возросшего объёма вычислительных операций.

3. При выполнении вычислений на троичной машине выявляется значительно меньше ошибок расхождения результатов, чем при выполнении тех же вычислений на двоичной машине.

5. Заключение

В статье представлены основные возможности пакета комплексной арифметики, созданного в ДССП-ТВМ для троичной виртуальной машины. Пояснены ключевые аспекты реализации этого пакета на языке ДССП-Т. Продемонстрированы примеры его применения для построения вычислительных программ в интерпретаторе ДССП/ТВМ.

На примерах программ, выполняющих преобразование Фурье для комплексных векторов, показано, что вычисления на троичной машине можно производить с лучшей точностью, чем аналогичные вычисления на двоичной машине.

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН «Проведение фундаментальных научных исследований (47 ГП)» по теме № FNEF-2022-0004 «Разработка архитектуры, системных решений и методов для создания микропроцессорных ядер и коммуникационных средств семейства систем на кристалле двойного назначения»,
Рег. № 122041100063-0.

Complex Arithmetic in DSSP for Ternary Machine

A.A. Burtsev

Abstract. The DSSP-TVM software complex created at the Research Laboratory of Ternary Informatics at the Faculty of Computational Mathematics and Cybernetics of Moscow State University can be used as an environment for developing and running programs for a ternary computer. Initially, it provided only operations of integer arithmetic. Then the DSSP-TVM was supplemented with a package of real floating point arithmetic operations. Now a package of complex arithmetic has been created on its basis. The article describes the main possibilities of the proposed package, as well as explains the most important aspects of its implementation in the DSSP-T language in the environment of the DSSP/TVM interpreter.

Keywords: ternary symmetric number system, ternary machine, ternary logic, ternary arithmetic, DSSP, floating point real arithmetic, complex arithmetic, rounding.

Литература

1. Кнут Д. Искусство программирования на ЭВМ. Т.2 Получисленные алгоритмы. М., Мир, 1977, п.4.1, 216-219.

2. Н.П. Брусенцов Н.П., Х. Рамиль Альварес. Троичные ЭВМ “Сетунь” и “Сетунь 70”. «Первая Международная конференция "Развитие вычислительной техники в России и странах бывшего СССР: история и перспективы" SORUCOM-2006», Россия, Петрозаводск, 3-7 июля 2006. Петрозаводск, Изд-во ПетрГУ, 2006, ч.1, 45-51.

3. Маслов С.П. Об одной возможности реализации троичных цифровых устройств. «Программные системы и инструменты» Т.12 (2011), 222-227.

4. Брусенцов Н.П. Заметки о троичной цифровой технике. «Вычислительная техника и вопросы кибернетики», Т.15 (1978), 145–155.

5. Владимирова Ю.С. Введение в троичную информатику: учебное пособие. М., АРГАМАК-МЕДИА; 2015.

6. А.А. Бурцев, В.А. Бурцев. О преимуществах троичных машин и эффективности троичных вычислений. «Труды НИИСИ РАН», Т. 10 (2020), № 3, 60–65.

7. Ким Г.Д., Воеводин В.В. Машинные операции с точки зрения математика. «Вычислительные методы и программирование», Т. 26 (1977), 31-35.

8. Бурцев А. А., Сидоров С. А. Троичная виртуальная машина и троичная ДССП. «Программные системы: теория и приложения» Т.6 (2015), №4, 29–97, http://psta.psiras.ru/read/psta2015_4_29-97.pdf.

9. Бурцев А.А., Сидоров С.А. История создания и развития ДССП: от "Сетуни-70" до троичной виртуальной машины. «Вторая Международная конференция “Развитие вычислительной техники и её программного обеспечения в России и странах бывшего СССР” SORUCOM-2011», Россия, В.Новгород, 12-16 сентября 2011. В.Новгород, Изд-во НовГУ, 2011, 83-88.

10. А.А Бурцев. Вещественная арифметика в ДССП для троичной машины. «Труды НИИСИ РАН», Т. 11 (2021), № 2, 33–41.

11. Бурцев А.А., Рамиль Альварес Х. Средства объектно-ориентированного программирования в ДССП. «Программные системы и инструменты. Тематический сборник №4», М., Изд-во факультета ВМК МГУ, 2003, 166-175.

12. А.А. Бурцев, М.А. Бурцев. ДССП для троичной виртуальной машины. «Труды НИИСИ РАН», Т. 2 (2012), № 1, 73–82.

13. А.А. Бурцев. Разработка собственных управляющих конструкций в среде ДССП для троичной машины. «Труды НИИСИ РАН», Т. 8 (2018), № 2, 52–64.

14. А.А. Бурцев. О возможности применения векторного сопроцессора для ускорения операции быстрого преобразования Фурье. «Труды НИИСИ РАН», Т. 5 (2015), № 2, 138–147.

О произведении множеств с единичной плотностью и его дополнении

Ю.Н. Штейников[1]

[1]ФГУ ФНЦ НИИСИ РАН, Москва, Россия, yuriisht@gmail.com

Аннотация. В статье S.Bettin; D. Koukoulopoulos, C. Sanna «A note on the natural density of product sets», Bull. Lond. Math. Soc. 53, No. 5, 1407-1413, 2021 было доказано, что множество произведений двух произвольных множеств целых чисел с единичной плотностью снова является множеством с плотностью единица. В этой же статье также были приведены верхнице оценки для так называемых соответствующих дополняющих множеств при некоторых специальных ограничениях. В настоящей работе доказываются аналогичные оценки при более слабых ограничениях..

Ключевые слова: плотность, подмножества, произведение

1. Введение

Всюду в этой статье A,B обозначают подмножества натуральных чисел. Стандартным образом обозначим через AB множество, которое является произведением множеств A и B, и определяется следующим образом

$$AB = \{ab : a \in A,\ b \in B\}$$

Исследования по изучению свойств произведения множеств имеет богатую историю и восходит к П. Эрдешем, его знаменитой проблеме о таблице умножения. Данный вопрос ставился так. Каков размер множества натуральных чисел, представимых в виде произведения ab, где элементы a, b принадлежат отрезку $[1, n]$, если n — достаточно большое число. Верно ли что множество таких произведений имеет размер по порядку меньший чем n^2? Данный вопрос был довольно подробно изучен самим П.Эрдешем и в дальнейшем Г.Тененбаумом и К.Фордом, установившим точный порядок роста этой величины.

Стоит отметить, что изучение произведения множеств довольно активно развивалось такими известными специалистами как Х.Силлеруело, О.Рамаре, С.Рамана, К.Санна и другими специалистами. Они рассматривали случайные множества A отрезка $[1, n]$, а также некоторый специальный класс множеств с нулевой плотностью и ставили задачу о размере множества произведений AA и частных A / A.

Перейдем к случаю бесконечных множеств A. Нам понадобится понятие плотности, которое определим ниже.

$$d(A) = \lim_{x \to \infty} \frac{|A \bigcap [1, x]|}{x}$$

Через $s(A)$ обозначим множество, являющееся дополнением множества A до множества всех натуральных чисел.

Н.Хегувари, П. Пач , Ф.Хеннекарт и поставили вопросы, один из которых звучит таким образом.

Вопрос 1. *Верно ли, что если* $d(A) = 1$ *, то отсюда следует, что* $d(AA) = 1$ *?*

В статье [1] был дан положительный ответ на данный вопрос и соответственно доказана такая теорема.

Теорема 1. *Если* $d(A) = 1$ *, то* $d(AA) = 1$.

В этой же статье это утверждение было количественно уточнено. А именно, когда

$$\left| s(A) \bigcap [1, n] \right| \le \frac{n}{\log^b n} \qquad (*)$$

для некоторого фиксированного $b \in (0,1)$.

Приведем утверждение, которое получилось у указанных авторов в данном случае.

Пусть для некоторого фиксированного $0 < b < 1$ *выполнено условие (*), тогда*

$$\left| \{ s(AA) \bigcap [1, n] \} \right| < n / (log\ n)^{c + o(1)},$$

где $c = b^2 / (1 + b)$

Из оценки видно, что если b близко к нулю, то оценка становится не такой эффективной и требует уточнения. Иными словами, какая будет оценка на порядок роста величины $s(AA)$ в случае, когда b близко к нулю. Для этого рассмотрим случай

$$| s(A) \bigcap [1, n] | < n / (log\ log\ n)^a \quad (**)$$

и пусть a>1 – некотрый фиксированный параметр.

Замечание. Условие $a > 1$ является чисто

техническим и нужно для проведения вычислений и оценок.

Будем основываться на аргументах и технике, предложенной в статье [1]. Мы получаем аналогичные оценки в вышеописанном случае.

Главным результатом настоящей статьи является следующее утверждение.

Теорема 2. *Пусть* $a > 1$ - *некоторое фиксированное число и выполнено соотношение (**) для множеств* A, B. *Тогда имеет место неравенство.*

$$\left|\left\{s(AB) \cap [1,n]\right\}\right| < n/\left(log\ log\ log\ n\right)^{-a+o(1)}.$$

Доказательство этой оценки опирается в основном на рассуждения и аргументы из вышеуказанной статьи [1] с учетом некоторых дополнительных соображений.

Замечание. Легко видеть, что Теорему 2 и все предыдущие утверждения можно доказывать для множеств $A = B$, так как в противном случае мы можем взять пересечение этих двух множеств, которое будет также иметь единичную плотность и будет при этом выполняться соотношение (**). Применение теоремы к этому множеству не изменит вид нужных нам оценок.

2. Предварительные сведения.

Потребуется некоторые обозначения и утверждения.

Для целого n, пусть $P^-(n)$ и $P^+(n)$ обозначают наименьший и наибольший простой делитель n. Если $P^+(n) \le y$ мы называем число n $y-$ гладким. Введем определение

$$\psi(x,y) = \left\{ n \le x,\ P^+(n) \le y \right\},$$

$$\varphi(x,y) = \left\{ n \le x,\ P^-(n) \le y \right\}.$$

Общеизвестны следующие утверждения.

Утверждение 1. *Для* $1 \le y \le x$ *справедливо*

$$|\varphi(x,y)| \le \mathrm{c}\, x/\ log\ y.$$

Утверждение 2. *Пусть* $2 \le y \le x$, $v = \log x / \log y$ *и* $e > 0$ - *произвольное фиксированное число и* $v \le y^{1-\varepsilon}$. *Тогда справедлива формула*

$$|\psi(x,y)| = x v^{-(1+o(1))v},\ v \to \infty.$$

Пусть натуральное $n = n_1 n_2$, где

$$P^+(n_1) \le y,\ P^-(n_2) \ge y.$$

Пусть $N(x,y,z)$ обозначает множество

$$\{n \le x : n_1 > z.\}$$

В статье [1] была получена верхняя оценка для $N(x,y,z)$. С помощью более тонких аргументов мы получаем более точную по порядку оценку на множество $N(x,y,z)$.

Утверждение 3. Пусть $y \le z \le x$, вещественное число u определяется равенством $u = \log z / \log y$. Пусть для произвольного фиксированного $e > 0$ выполнено

$$log\ x / log\ y \le y^{1-e}.$$

Тогда справедлива оценка

$$\left|N(x,y,z)\right| \prec x\ exp\left\{-\left(1+o(1)\right) u\ log\ u\right\},$$

при $u \to \infty$.

Доказательство. Справедливо следующее соотношение

$$\left|N(x,y,z)\right| = \sum_{z \le n \le x,\ P^+(n) \le y} \varphi(x/n, y).$$

Последнюю сумму разобьем на 2 части.

$$\sum_{z \le n \le x/y,\ P^+(n) \le y} \varphi(x/n, y) +$$

$$+ \sum_{x/y \le n \le x,\ P^+(n) \le y} \varphi(x/n, y).$$

Последнее же слагаемое оценивается через множество $y-$гладких чисел, не превосходящих x, то есть величиной

$$x\ exp\left\{-\left(1+o(1)\right) u\ log\ u\right\}.$$

Разберемся с первой суммой.

$$\sum_{z \le n \le x/y,\ P^+(n) \le y} \varphi(x/n, y) \le$$

$$\le (x/log\ y) \sum_{z \le n \le x/y,\ P^+(n) \le y} 1/n.$$

Разберемся с последней суммой, применив преобразование Абеля.

$$\sum_{z \le n \le x/y,\ P^+(n) \le y} 1/n \le$$

$$\le \left(\left|\psi(x/y, y)\right| - |\psi(z,y)|\right)/(x/y) +$$

$$+ \int_z^{x/y} \left\{\left|\psi(t,y)\right| - |\psi(z,y)|\right\}/t^2 dt$$

Делаем замену переменной в интеграле $t \to h$, которая связывает переменные взаимно-однозначным соотношением $t = y^h$. Интеграл преобразуется к виду

$$(log\ y\ )\ *$$
$$*\ \int_{u}^{log\ x/log\ y} \left\{\left|\psi\left(y^h, y\right)\right|\ -\ \left|\psi\left(y^u, y\right)\right|\right\} / y^h\ dh$$

Подставляя верхние оценки на множество гладких чисел (Утверждение 2) и выполняя несложные вычисления, мы получаем требуемое. Утверждение 3 доказано.

3. Доказательство основного результата.

Будем предполагать (как и ранее было указано), что $A = B$. Введем параметр $u = u(x) > 1,\ y\ = y\left(x\right)\ < x,\ z\ = y^u$.

Рассмотрим множество

$$N_1 = \left\{n < x :\ n_1 < z\right\}.$$

Исходя из оценок на множество $N\left(x, y, z\right)$ легко видеть, что

$$\left|\left[1, x\right] \setminus\ N_1\right|\ =$$
$$=\ \left|N\left(x, y, z\right)\right|\ \le$$
$$\le\ x\ exp\ \left\{-\ \left(1+o\left(1\right)\right)\ u\ log\ u\ \right\}.$$

Далее будем рассматривать только лишь числа $n \in N_1\left(x, y, z\right)$ преставимые в виде $a_1 a_2$, где $a_1\ , a_2 \in A$. Сделаем следующее наблюдение. Если n не принадлежит множеству AA, то либо n_1 не принадлежит множеству A, либо n_2 не принадлежит A. Значит множество таких элементов не превосходит $S1\ +\ S2$, где по определению

$$S_1\ =\ \left|\left\{n\ :\ n_1\ \notin A\ \right\}\right|,$$
$$S_2 =\ \left|\left\{n\ :\ n_2\ \notin\ A\right\}\right|.$$

Оценим сначала сверху S_1, применяя частное суммирование и огрубляя суммирование по всем числам из дополнения множества A:

$$S_1\ \le\ \sum_{m\ \le z,\ m\ \notin A} \varphi\left(x/m, y\right)\ \le$$
$$\le x/\log y \sum_{m\ \le z,\ m\ \notin A} \left(1/m\right) \le$$
$$\le ux/\left(log\ \ log\ y\right)^a.$$

Оценим теперь S_2 сверху. Сперва обозначим

$$s(A)\left(t\right) = \left|\left\{\left(\mathrm{N} \setminus A\right)\ \cap\ \left[1, t\right]\right\}\right|.$$

Поэтому мы можем оценить величину S_2 таким образом

$$S_2\ \le\ \sum_{d:\ d\ \le z,\ P+(d)\ \le y} \mathrm{s}(A)\left(x/d\right)\ \le$$
$$\le x \log y/\left(\log \log x\right)^a\ .$$

Следовательно множество исключений складывается из величин $S1$, $S2$ и $N\left(x, y, z\right)$. Иными словами имеет место неравенство $\left|s\left(AB\right) \cap \left[1, x\right]\right|\ <\ \left|N\left(x, y, z\right)\right|\ +\ S_1\ +\ S_2$. Подбирая должным образом параметры $u =\ u\left(x\right),\ y =\ y\left(x\right)$, мы приходим к требуемой оценке. Терема доказана..

Публикация выполнена в рамках государственного задания ФГУ ФНЦ НИИСИ РАН по теме № FNEF-2022-0011.

On the Product of Sets with Density 1 and Its Addition

Y. N. Shteinikov

Abstract. S. Bettin, D. Koukoulopuos and C. Sanna proved in 2021 that the set of products of two arbitrary sets of integer with unit density is again a set with density of one. Article 3 of the authors also provides the top grades for the so-called corresponding complement sets under some special constraints. This article provides similar estimates for weaker restrictions.

Keywords: density, subsets, product

Литература

1. S.Bettin; D. Koukoulopoulos, C. Sanna A note on the natural density of product sets. Bull. Lond. Math. Soc. 53, No. 5, 1407-1413 (2021).

Подписано в печать 06.12.2022 г.
Формат 60x90/8
Печать цифровая. Печатных листов 8.25
Тираж 100 экз. Заказ № 1242

Отпечатано в ФГБУ «Издательство «Наука»
121099, Москва, Шубинский пер., 6