компьютер в учебном процессе

Абрамян Михаил Эдуардович

УДК 004.42+37.09

ИСПОЛЬЗОВАНИЕ ЭЛЕКТРОННОГО ЗАДАЧНИКА ПО СТРОКОВЫМ АЛГОРИТМАМ БИОИНФОРМАТИКИ[1]

Аннотация

Статья посвящена особенностям использования электронного задачника Programming Taskbook for Bioinformatics при изучении алгоритмов неточного сравнения строк. В качестве примера рассматривается серия заданий, связанная с алгоритмом глобального выравнивания. Приводится подробное описание процесса выполнения одного из заданий с использованием электронного задачника и дается обзор других заданий, входящих в эту серию.

Ключевые слова: электронный задачник, строковые алгоритмы поиска и неточного сопоставления, биоинформатика.

1. ВВЕДЕНИЕ

Настоящая статья, как и ранее опубликованная работа [1], посвящена электронному задачнику по строковым алгоритмам биоинформатики Programming Taskbook for Bio (PT for Bio). В [1] были рассмотрены особенности применения электронных задачников при изучении сложных алгоритмов, кратко описан универсальный задачник Programming Taskbook, на базе которого разработан задачник по биоинформатике, дана общая характеристика и описан состав задачника PT for Bio, а также приведен пример выполнения простого задания из группы Match, связанной с алгоритмами поиска подстрок. Объем работы не позволил дать развернутое описание особенностей задачника и более подробно рассмотреть группу Align, посвященную алгоритмам неточного сопоставления строк. В то же время, именно алгоритмы неточного сопоставления строк активно применяются в биоинформатике [2] и не так широко известны, как алгоритмы поиска, поэтому применение электронного задачника для их изучения представляет особый интерес. Предлагаемая статья призвана восполнить этот пробел. В ней содержится подробное описание основных этапов выполнения типового задания из группы Align, а также дается обзор серии, в которую входит рассмотренное задание. Чтобы продемонстрировать универсальный характер задачника, в качестве языка программирования в настоящей работе выбран язык C++ (в то время как в примере из [1] использовался язык Python).

Рассмотренные примеры заданий и особенности их выполнения позволяют более полно оценить возможности задачника и те преимущества, которые он предоставляет при изучении алгоритмов.

[1] Работа выполнена в рамках ФЦП «Научные и научно-педагогические кадры инновационной России» (госконтракт № 14.740.11.0006).

2. ВЫБОР УЧЕБНОГО ЗАДАНИЯ И СОЗДАНИЕ ПРОЕКТА-ЗАГОТОВКИ

В большинстве алгоритмов неточного сравнения строк, описанных в группе Align, можно выделить два этапа: *этап восходящей рекурсии*, в ходе которого строится вспомогательная матрица, и *этап обратного прохода* по построенной матрице. После выполнения этапа восходящей рекурсии можно определить числовую характеристику, являющуюся мерой близости сравниваемых строк. Это либо варианты *редакционного расстояния* (задания Align5, Align12, Align18), либо варианты *сходства*, связанного с глобальным выравниванием (Align31, Align38, Align43, Align48) или выравниванием отдельных частей анализируемых строк (Align53, Align59). В алгоритмах нахождения наибольшей общей подстроки или наибольшей общей подпоследовательности двух строк после построения вспомогательной матрицы можно определить *длину* общей подстроки (Align63) или, соответственно, общей подпоследовательности (Align67).

В качестве примера заданий указанного типа рассмотрим задание Align31, посвященное реализации алгоритма восходящей рекурсии для нахождения сходства двух строк, связанного с глобальным оценочным выравниванием.

Align31. Даны строки S_1 и S_2. Кроме того, дано число m (< 10) – размер алфавита, состоящего из первых строчных латинских букв, и матрица оценок (структура матрицы описана в Align29). Для эффективного вычисления сходства строк $V(|S_1|, |S_2|)$ по формулам из Align30 можно использовать *метод восходящей рекурсии*, основанный на последовательном вычислении элементов вспомогательной матрицы V размера $(|S_1| + 1) \times (|S_2| + 1)$ (действия при заполнении матрицы V аналогичны действиям при заполнении матрицы D, описанным в Align5). Элемент, расположенный в последней строчке и последнем столбце матрицы V, определяет значение сходства данных строк. Используя метод восходящей рекурсии, найти матрицу V и вывести элементы ее двух последних строчек:

$$V_{n-1,\,j}, j = 0, \ldots, |S_2|, V_{n,\,j}, \; j = 0, \ldots, |S_2|,$$

где $n = |S_1|$.

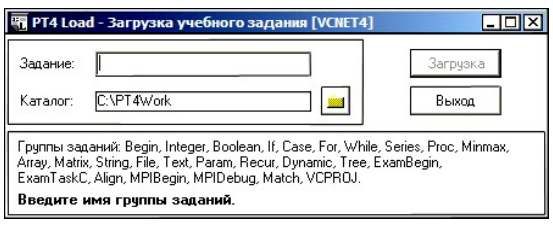

Рис. 1. Окно программного модуля PT4Load

В формулировке задания Align31 содержится ряд ссылок на предыдущие задания, в которых вводятся новые понятия, даются формулы или описываются действия, используемые при реализации требуемого алгоритма. Наличие большого числа подобных ссылок характерно для многих заданий группы Align. Это позволяет избежать дублирования определений и формул и тем самым уменьшает размеры формулировок заданий. Кроме того, ссылки дают возможность легко установить связи между рассматриваемыми «родственными» алгоритмами. Заметим, что в формулировках заданий мы, следуя переводу книги [2], называем строчками *матричные* (то есть табличные) строки (англ. row), оставляя термин «строка» (string) для *текстовых* строк.

Первым шагом при выполнении задания с применением электронного задачника PT for Bio является создание программы-заготовки для выбранного задания. Для этого студент должен перейти в свой рабочий каталог, связанный с задачником, и вызвать программный модуль PT4Load с помощью содержащегося в каталоге ярлыка Load. На экране появится окно модуля, в котором будут перечислены все доступные группы заданий (см. рис. 1).

В списке присутствуют группы, входящие в универсальный задачник Programming Taskbook и связанные с различными разделами базового курса программирования (тему, которой посвящена та или иная группа, легко определить по имени группы; например, группы Array и Matrix содержат задания на обработку одномерных и двумер-

ных массивов, группы File и Text – на обработку двоичных и текстовых файлов, группы ExamBegin и ExamTaskC связаны с заданиями ЕГЭ по информатике). Список групп увеличивается при установке дополнений – специализированных задачников. Например, после установки задачника по параллельному программированию PT for MPI [3] в списке появятся группы MPIBegin и MPIDebug, а после установки задачника по биоинформатике – группы Match и Align.

Модуль PT4Load позволяет выбрать не только задание, но и программную среду, в которой это задание будет выполняться. Имя текущей программной среды и номер ее версии указывается в заголовке окна модуля. Приведенный рисунок соответствует настройке, при которой текущей средой является среда Microsoft Visual Studio .NET 2010 для языка C++. Для выбора другой среды достаточно вызвать контекстное меню модуля PT4Load и выбрать нужную среду из появившегося списка. Следует отметить, что в любых средах языка C++, поддерживаемых задачником, действия при выполнении заданий будут практически одинаковыми.

После ввода допустимого имени задания (в нашем случае «Align31») кнопка «Загрузка» станет доступной; при ее нажатии будет создан проект-заготовка для указанного задания, автоматически запустится выбранная среда программирования и в нее загрузится созданный проект.

Программный модуль PT4Load можно использовать не только для создания, но и для загрузки уже существующего проекта, связанного с требуемым заданием.

Проект-заготовка на языке C++ состоит из нескольких файлов, однако в редактор среды будет загружен один файл с именем Align31.cpp, поскольку именно в этом файле надо запрограммировать решение задачи. В листинге 1 приводится начальная часть файла Align31.cpp (завершающая часть содержит описание стартовой функции WinMain, которую при выполнении задания изменять не требуется). Файл содержит директивы подключения необходимых заголовочных файлов (в частности, файла pt4.h с объявлениями вспомогательных функций задачника) и описание функции Solve, с вызова которой начинается процесс выполнения задания (вызов функции Solve производится в try-блоке функции WinMain, что позволяет задачнику перехватывать и обрабатывать ошибки времени выполнения). Первым оператором функции Solve является оператор вызова функции Task, инициализирующей задание Align31.

3. ОКНО ЗАДАЧНИКА

Созданная программа уже может быть запущена на выполнение; в результате на экране появится окно задачника (рис. 2). С помощью этого окна студент может ознакомиться с формулировкой задания, вариантом входных данных и соответствующими этому варианту правильными результатами.

Кроме двух исходных строк S_1 и S_2 и числа m, определяющего количество начальных букв латинского алфавита, которые могут входить в исходные строки, в задании приводятся элементы *матрицы оценок s*, определяющие относительный «вес» пар символов при их выравнивании. Определение матрицы оценок приводится в задании Align29. Это квадратная матрица порядка $m + 1$; ее строчки и столбцы связываются с различными символами алфавита, используемого в исходных строках, а также с символом пробела (с этим символом связывается последняя строчка и последний столбец матрицы оценок). Во всех заданиях предполагается, что матрица оценок является симметричной, поэтому в наборе исходных данных приводится только верхняя треугольная часть этой матрицы. Для большей наглядности строчки и столбцы матрицы оценок снабжаются заголовками, содержащими те сим-

Листинг 1

```
#include <windows.h>
#pragma hdrstop
#include "pt4.h"
void Solve()
{
    Task("Align31");
}
```

волы, которым они соответствуют. В разделе результатов приведены значения элементов двух последних строчек матрицы V, которая должна быть получена после выполнения этапа восходящей рекурсии.

Запуск программы с заданием считается *ознакомительным*, поскольку в программе не выполняются действия по вводу-выводу данных.

Приведенное на рис. 2 стандартное окно задачника ориентировано, прежде всего, на задания, для которых формулировки и наборы данных занимают не более пяти экранных строк. Таково большинство заданий, входящих в базовый вариант задачника. Однако задания, связанные с реализацией сложных алгоритмов, подобные рассматриваемому заданию Align31, включают существенно более развернутые формулировки (которые, кроме того, нередко снабжаются примечаниями) и большие наборы данных. При выполнении таких заданий удобнее использовать альтернативный вариант окна, размеры которого определяются размерами формулировки задания и наборов данных. Для переключения в альтернативный режим достаточно нажать кнопку быстрого доступа «Режим», расположенную в правом верхнем углу окна. Вид альтернативного окна для задания Align31 приведен на рис. 3.

При выходе из программы задачник запоминает выбранный режим окна и при последующем запуске автоматически восстанавливает его.

4. ДОПОЛНИТЕЛЬНЫЕ РЕЖИМЫ ЗАПУСКА ПРОГРАММЫ С УЧЕБНЫМ ЗАДАНИЕМ

Ознакомление с деталями алгоритма, которому посвящено задание, часто требует обращения к другим заданиям, упомянутым в формулировке. Для этого можно, например, временно изменить имя задания, указанное в качестве параметра процедуры Task, и повторно запустить программу. В результате в окне задачника будет выведена информация, связанная с новым заданием. Однако удобнее запустить программу с заданием в *демонстрационном режиме*. Для этого достаточно добавить к имени задания в функции Task символ «?» (например, «Align31?»). При демонстрационном запуске проверка правильности решения не производится, однако появляется возможность просматривать все задания, входящие в указанную группу, используя дополнительные кнопки окна задачника (см. рис. 4). Кроме того, для каждого задания можно просматривать различные варианты исходных данных и соответствующие им результаты.

Другим удобным способом ознакомиться со всеми заданиями группы является генерация html-страницы с формулировками всех заданий. Для этого достаточно в качестве параметра функции Task указать имя группы, дополненное символом «#» (в нашем случае «Align#»). После генерации html-страницы автоматически запускается

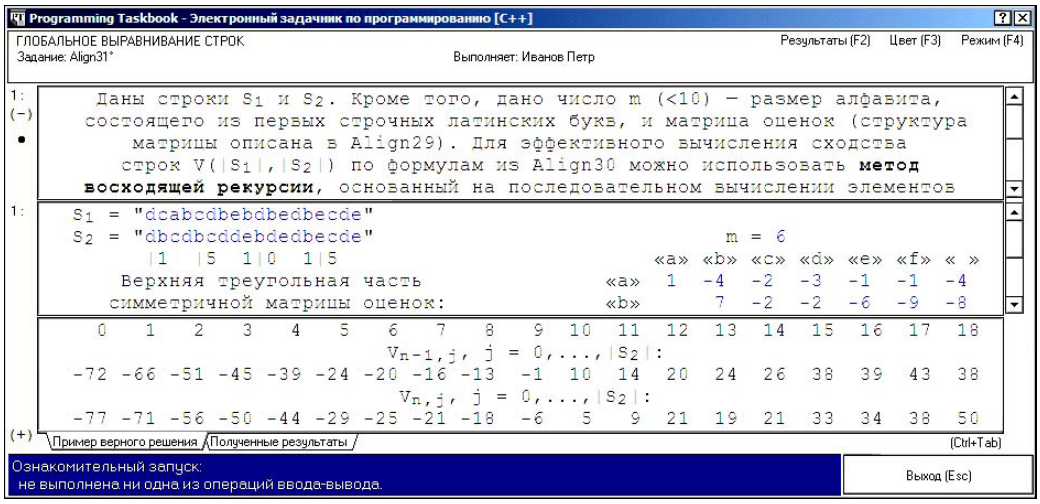

Рис. 2. Ознакомительный запуск задания Align31 (стандартный режим окна)

интернет-браузер, назначенный по умолчанию для данного компьютера, и в него загружается эта страница. Помимо формулировок учебных заданий html-страница может содержать *преамбулу*, в которой дается общее описание группы, описываются используемые термины и обозначения, а также налагаются ограничения на исходные данные, которые действуют для всех (или большинства) заданий группы.

Пользуясь отмеченными возможностями, студент должен ознакомиться со структурой матрицы оценок, описанной в задании Align29, а из формулировки задания Align30 получить рекуррентные формулы для элементов матрицы V (задание Align30 приведено на рис. 4).

Идея метода восходящей рекурсии описывается в задании Align5 (в применении к матрице *редакционных расстояний*). Она состоит в последовательном вычислении элементов матрицы по строчкам после предварительного нахождения базовых элементов матрицы, расположенных в ее нулевой строчке и столбце. При этом на этапе вычисления элемента матрицы, расположенного в *i*-й строчке и *j*-м столбце, *уже известны* все три элемента, входящие в формулу для $V_{i,j}$, а именно элементы, расположенные выше, левее и одновременно выше и левее элемента $V_{i,j}$.

5. ОСОБЕННОСТИ ВВОДА-ВЫВОДА

Усвоив алгоритм в полном объеме, следует приступить к его программной реализации. Прежде всего, необходимо определить структуры для хранения исходных, промежуточных данных и результатов, а также организовать ввод-вывод данных.

При выполнении заданий с применением электронного задачника программа студента получает исходные данные непосредственно от задачника. Это, во-первых, позволяет существенно *ускорить* процесс отладки, так как не требует ввода данных с клави-

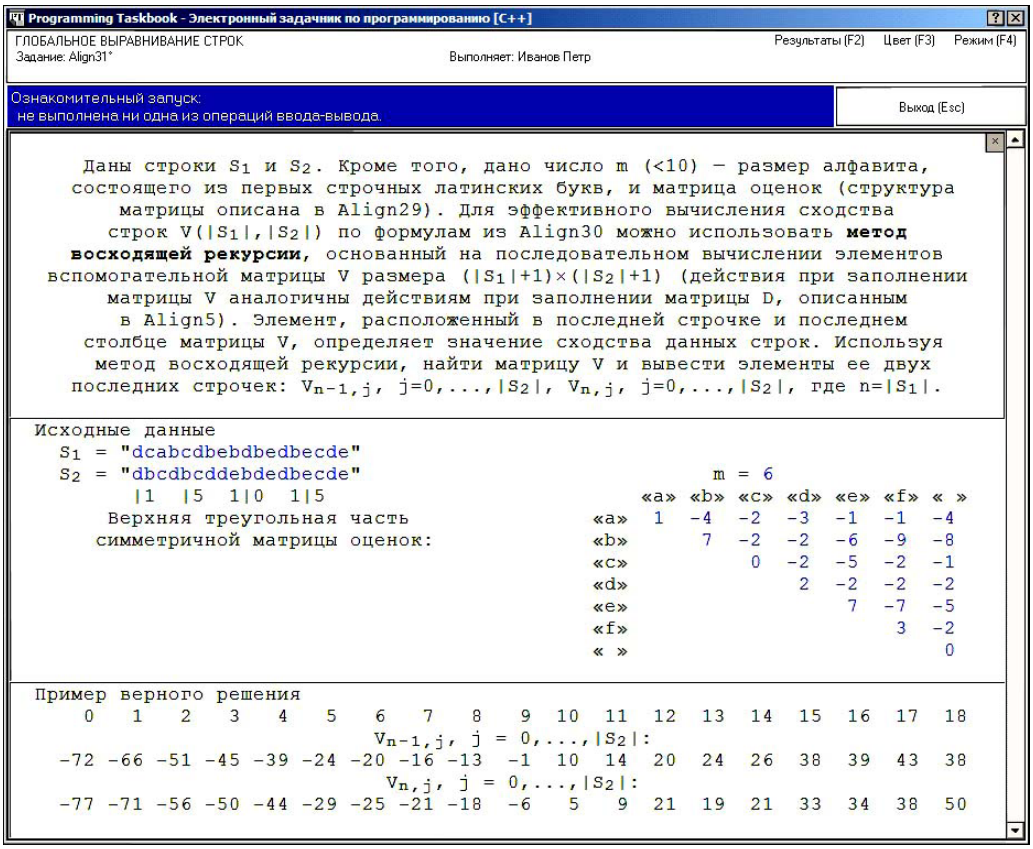

Рис. 3. Ознакомительный запуск задания Align31 (альтернативный режим окна)

атуры или их предварительного занесения в файл, и, во-вторых, повышает *надежность* отладки, поскольку задачник при каждом тестовом запуске генерирует новый набор исходных данных. Исходные данные генерируются с применением датчика случайных чисел и с учетом особенностей выполняемого задания; в частности, если в алгоритме решения требуется предусмотреть обработку каких-либо особых случаев, то можно гарантировать, что при некоторых тестовых испытаниях программе будут предложены наборы, соответствующие этим особым случаям.

Для получения данных, подготовленных задачником, программа студента должна использовать специальные средства ввода, аналогичные стандартным средствам соответствующего языка программирования. Например, для языков Pascal и Visual Basic предусмотрен набор *процедур*, каждая из которых предназначена для ввода одного элемента данных определенного типа. В варианте задачника для языка C++ предусмотрен специальный *поток* pt, объявленный в заголовочном файле pt4.h.

Вывод результатов должен выполняться таким образом, чтобы они были доступны задачнику для проверки. С этой целью используются специальные средства вывода. Для языка C++ вывод может осуществляться с помощью того же потока pt (который, таким образом, является потоком ввода-вывода). Результаты, полученные программой,

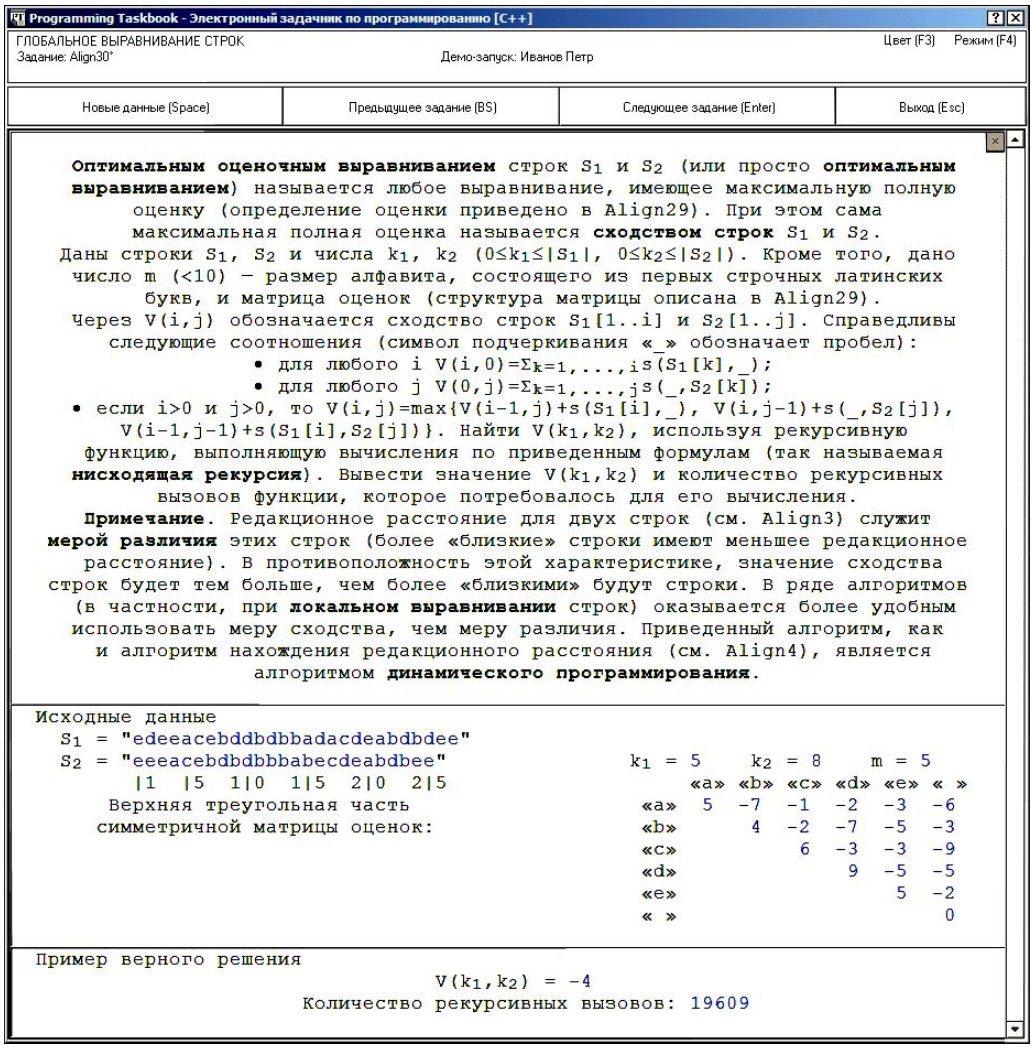

Рис. 4. Демонстрационный запуск программы для группы Align

отображаются на экране, причем в случае ошибочного решения студент может сравнить свои результаты с правильными результатами, которые также отображаются в соответствующем разделе окна задачника.

Контроль за организацией ввода и вывода достаточно просто автоматизировать, поэтому задачник в состоянии самостоятельно выявлять различные ошибки ввода-вывода. На начальном этапе выполнения задания студенты часто допускают подобные ошибки, которые, разумеется, будут приводить к неправильной работе всей программы. Автоматический контроль за ошибками ввода-вывода существенно упрощает поиск таких ошибок и обеспечивает их быстрое исправление. Задачник в состоянии диагностировать три основных типа ошибок ввода-вывода: это ошибки, связанные с вводом-выводом *недостаточного* количества данных, с попыткой ввода-вывода *лишних* данных и с попыткой ввода-вывода данных *неверного типа*.

В качестве иллюстрации описанных выше особенностей задачника рассмотрим вариант функции **Solve**, в котором вводится часть исходных данных, а именно исходные строки **s1** и **s2** и размер **m** матрицы оценок (листинг 2).

Листинг 2

```
void Solve()
{
    Task("Align31");
    string s1, s2;
    int m;
    pt >> s1 >> s2 >> m;
}
```

Запуск данного варианта программы уже не будет считаться ознакомительным, поскольку программа выполняет действия по вводу данных. Однако вводятся не все исходные данные, поэтому задачник выведет соответствующее сообщение об ошибке (см. рис. 5). Для того чтобы еще более упростить для студента поиск и исправление подобных ошибок, в окне задачника дополнительно отображается панель индикаторов ввода-вывода, на которой в текстовом и графическом виде выводится информация о количестве введенных и выведенных элементов данных. Кроме того, на этой панели располагается индикатор прогресса выполнения задания, который будет заполняться при проведении успешных тестовых испытаний. Рисунок демонстрирует еще одну возмож-

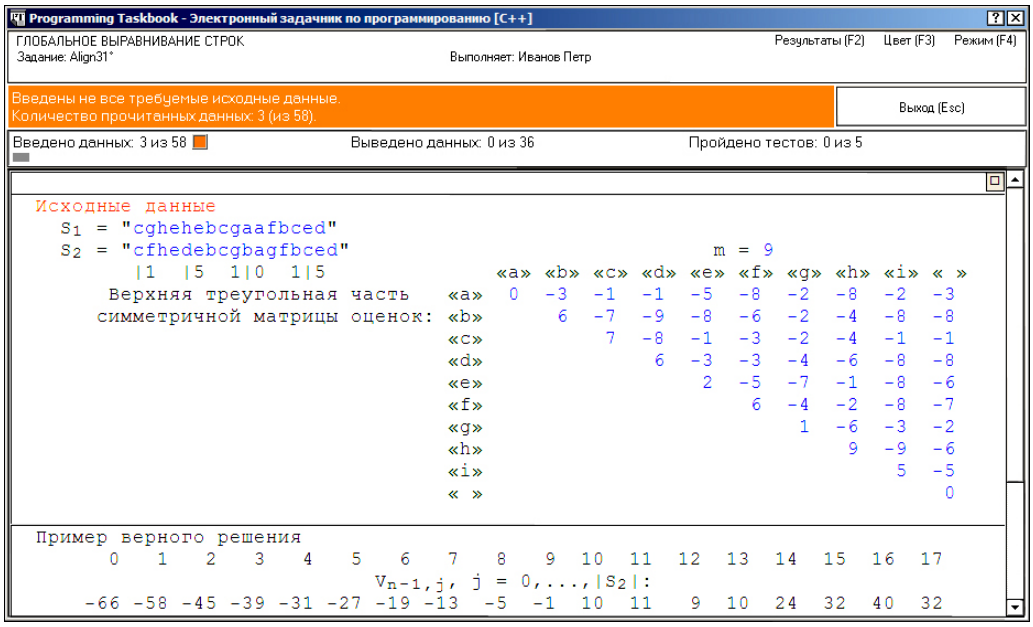

Рис. 5. Окно задачника с сообщением об ошибке ввода

ность окна задачника: для уменьшения размеров окна можно временно скрыть раздел с формулировкой заданий, выполнив щелчок мышью на маркере, расположенном в правом верхнем углу этого раздела.

6. ТЕСТИРОВАНИЕ ПРАВИЛЬНОГО РЕШЕНИЯ

Мы не будем приводить полный вариант программы с правильным решением задачи Align31, поскольку любой такой вариант имеет достаточно большой размер, и, кроме того, анализ подобного решения не выявит дополнительных особенностей, связанных с использованием электронного задачника. Примеры решений нескольких типовых задач из групп Match и Align (в том числе задачи Align31), причем с использованием различных языков программирования, приводятся в разделе «PT for Bio» сайта электронного задачника [4]. Отметим лишь, что разработанный вариант решения считается правильным только в случае, если он успешно пройдет серию тестовых испытаний, на каждом из которых задачник предложит для обработки новый вариант исходных данных. При неудачном тестовом испытании отсчет успешных тестов начинается заново. На рис. 6 приводится вид окна задачника после четвертого тестового испытания.

При успешном прохождении требуемого числа испытаний задание считается выполненным, и информация об этом заносится в файл результатов. В задачнике предусмотрены средства, позволяющие заносить в файл результатов информацию о выполнении задания только после того как преподаватель просмотрит текст программы с решением задачи.

7. ПРИМЕР СЕРИИ УЧЕБНЫХ ЗАДАНИЙ

В [1] была отмечена важная возможность, доступная при изучении сложных алгоритмов с помощью электронного задачника: подобное изучение можно организовать в виде выполнения *серии заданий*, каждое из которых посвящено новым понятиям, используемым в алгоритме, или реализации какого-либо этапа алгоритма.

В качестве примера подобной серии можно привести серию, включающую рассмотренное ранее задание Align31. Эта серия посвящена алгоритму нахождения сходства строк и построения оптимального оценочного выравнивания и включает 9 зада-

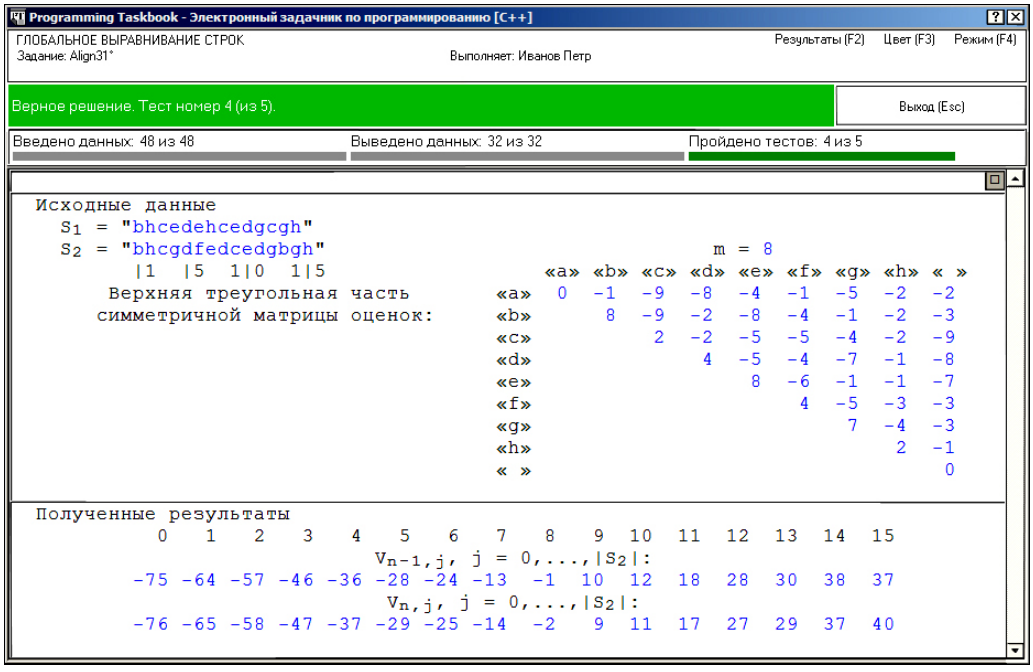

Рис. 6. Окно задачника с сообщением об успешном тестовом испытании

ний: Align29–Align37. При обсуждении задания Align31 уже упоминались вводные задания этой серии Align29 и Align30: в первом из них вводится в рассмотрение матрица оценок и дается определение полной оценки выравнивания (в задании требуется вычислить оценку выравнивания для данных двух строк, причем сам задачник предоставляет некоторое выравнивание для этих строк, а также матрицу оценок); во втором задании вводится понятие оптимального оценочного выравнивания и приводятся рекуррентные формулы для его вычисления на основе построения вспомогательной матрицы V (в задании требуется реализовать вычисление некоторого элемента матрицы V, используя простой, но неэффективный алгоритм *нисходящей рекурсии*; вид окна задачника для этого задания приведен на рис. 4).

Продолжает серию набор заданий, посвященный реализации алгоритма восходящей рекурсии. Кроме Align31 в этот набор входит задание Align32, в котором требуется модифицировать алгоритм восходящей рекурсии таким образом, чтобы он позволял найти не только значение сходства строк, но и количество различных вариантов оптимального выравнивания.

Следующие три задания серии посвящены второму этапу алгоритма: *обратному проходу* по матрице, полученной методом восходящей рекурсии. Обратный проход используется для нахождения какого-либо варианта (или всех вариантов) оптимального редакционного предписания (Align7–9, Align14–15, Align20–21), глобального выравнивания (Align33–35, Align39–40, Align44–45, Align49–50), приближенного вхождения одной строки в другую (Align55–56), локального выравнивания (Align60–61), наибольшей общей подстроки (Align64–65), наибольшей общей подпоследовательности (Align68).

Приведем формулировку задания Align33, в которой описывается алгоритм обратного прохода.

Align33. Даны строки S_1 и S_2, число m (< 10) – размер алфавита, состоящего из первых строчных латинских букв, и матрица оценок (структура матрицы описана в Align29). Кроме того, дана матрица V, полученная методом восходящей рекурсии (см. Align31). Известно, что имеется единственный вариант оптимального выравнивания для S_1 и S_2. Требуется найти это выравнивание (A_1, A_2), выполняя *обратный проход* по матрице V, начиная с ее правого нижнего элемента и заканчивая левым верхним элементом; при этом для каждого элемента $V_{i,j}$ при $i > 0$ и $j > 0$ перемещение выполняется в тот из элементов $V_{i,j-1}$, $V_{i-1,j}$, $V_{i-1,j-1}$, на котором достигается максимум в формуле для $V(i, j)$ (см. Align30); для элементов $V_{0,j}$ перемещение всегда выполняется влево, а для элементов $V_{i,0}$ – вверх. Перемещение влево соответствует вставке пробела в строку A_1 и текущего элемента S_2 в строку A_2, перемещение вверх – вставке пробела в строку A_2 и текущего элемента S_1 в строку A_1, перемещение по диагонали влево и вверх соответствует вставке в A_1 и A_2 текущих элементов S_1 и S_2 соответственно (символы в строках S_1 и S_2 перебираются с конца, строки A_1 и A_2 заполняются от конца к началу). Вывести найденные строки A_1 и A_2.

Важной особенностью этого задания (как и всех перечисленных выше заданий, связанных с реализацией обратного прохода) является то, что в нем *не требуется* предварительно выполнять этап восходящей рекурсии, поскольку матрица V, необходимая для реализации обратного прохода, предоставляется в составе набора исходных данных. Следует также обратить внимание на дополнительное условие задания: предполагается, что имеется *единственный вариант* оптимального выравнивания; таким образом, при реализации обратного прохода каждый шаг определяется однозначно, что упрощает алгоритм. Все необходимые предварительные действия, связанные с подбором строк, допускающих единственное выравнивание, и с вычислением вспомогательной матрицы V, выполняются самим задачником.

Два следующих задания, посвященных обратному проходу, являются более сложными. В Align34 уже не предполагается, что оптимальное выравнивание является един-

ственным, и определяется порядок действий в случае, если при обратном проходе на каком-либо шаге возможны перемещения более чем в один из соседних элементов. Требуется найти тот вариант оптимального выравнивания, который соответствует описанному порядку действий. Наконец, в Align35 ставится наиболее общая задача: найти все варианты оптимального выравнивания и вывести их в определенном порядке.

Завершают серию *итоговые задания*, в которых требуется реализовать алгоритм оптимального выравнивания в полном объеме; при этом в Align36 требуется найти один из вариантов выравнивания, а в Align37 – все варианты.

Поскольку серия состоит из групп однотипных заданий – ознакомительных (Align29–Align30), связанных с этапом восходящей рекурсии (Align31–Align32), связанных с этапом обратного прохода (Align33–Align35) и итоговых (Align36–Align37), – преподаватель может подготовить варианты индивидуальных заданий, в каждый из которых будет входить по одному заданию из каждой группы. В состав комплекса Teacher Pack – дополнения к электронному задачнику, предназначенного для преподавателя [5], – входит *конструктор вариантов*, позволяющий автоматически генерировать подобные варианты индивидуальных заданий.

Литература

1. *Абрамян М. Э.* Реализация электронного задачника по строковым алгоритмам биоинформатики // Компьютерные инструменты в образовании, 2012. № 2. С. 43–52.

2. *Гасфилд Д.* Строки, деревья и последовательности в алгоритмах: Информатика и вычислительная биология. СПб.: БХВ-Петербург, 2003.

3. *Абрамян М. Э.* Электронный задачник по параллельному программированию на основе технологии MPI // Компьютерные инструменты в образовании, 2011. № 6. С. 47–54.

4. Электронный задачник Programming Taskbook / http://ptaskbook.com/ (дата обращения: 30.06.2012).

5. *Абрамян М. Э.* Использование специализированного программного обеспечения для преподавателя при организации и проведении лабораторных занятий по программированию // Информатика и образование, 2011. № 5. С. 78–80.

Abstract

We discuss essential features of the Programming Taskbook for Bioinformatics related to the study of approximate string matching algorithms. For that purpose we give a review of one group of educational tasks devoted to the global string alignment. We also provide step-by-step solution to a typical training task and briefly outline some other tasks in the group.

Keywords: educational software, exact and approximate string matching algorithms, bioinformatics.

Абрамян Михаил Эдуардович,
кандидат физико-математических наук, доцент кафедры алгебры и дискретной математики Южного федерального университета,
mabr@math.sfedu.ru