Использование электронного задачника при решении задач ЕГЭ на обработку файлов

М. Э. Абрамян

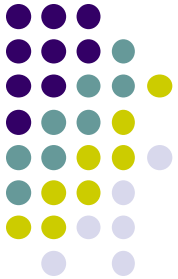
Южный федеральный университет,
Институт математики, механики
и компьютерных наук им. И. И. Воровича

Университет МГУ-ППИ в Шэньчжэне,
факультет вычислительной математики
и кибернетики

mabr@sfedu.ru, m-abramyan@yandex.ru

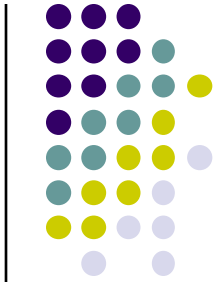
Всесоюзная научно-методическая конференция
«Использование системы программирования PascalABC.NET в обучении программированию»
(Ростов-на-Дону, 12–13 ноября 2021 г.)

Задачи ЕГЭ-2021, связанные с разработкой программ



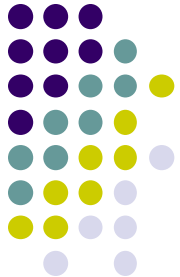
- При переходе на компьютерное тестирование **количество задач на разработку программ увеличилась**
- В большинстве таких задач для получения ответа необходимо обработать **текстовый файл**
- Трудности при организации практических занятий по подготовке к ЕГЭ:
  - необходимо использовать **набор однотипных заданий**, который позволяет проверить уровень усвоения изученных алгоритмов
  - необходимо иметь **файлы с исходными данными** для каждого задания
  - необходимо **проверить правильность ответа**

Задачник Programming Taskbook for Exam



- Является расширением универсального задачника Programming Taskbook, входит в состав варианта задачника для системы PascalABC.NET
- Включает три группы учебных заданий:
  - **ExamBegin** – 130 задач на освоение базовых алгоритмов
  - **ExamCheck** – 30 задач на исправление программы (данная группа может считаться *устаревшей*, так как подобные задачи при компьютерном тестировании не используются)
  - **ExamTaskC** – 155 задач повышенного уровня сложности

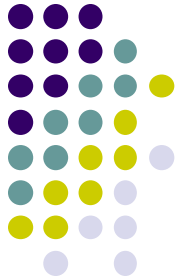
Состав группы ExamBegin



Темы задач группы ExamBegin в основном соответствуют списку типовых алгоритмов, приведенному в **кодификаторе ЕГЭ по информатике и ИКТ** 2021 г. (за исключением рекурсивных алгоритмов, алгоритмов приближенного решения уравнений и алгоритмов приближенного вычисления длин и площадей):

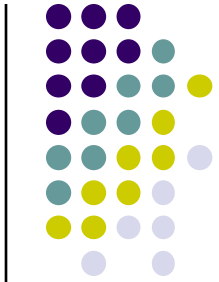
- **Условные операторы и циклы** (17 задач)
- **Формирование массивов** (12 задач)
- **Анализ одномерных массивов** (15 задач)
- **Минимумы и максимумы** (14 задач)
- **Анализ двумерных массивов** (10 задач)
- **Преобразование массивов** (16 задач)
- **Обработка текстовых данных** (16 задач)
- **Проверка делимости и выделение цифр из целых чисел** (8 задач):
- **Пары и тройки элементов массива** (14 задач)
- **Обработка статистических данных** (8 задач)

Состав группы ExamTaskC



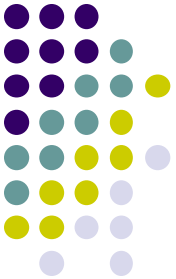
- Группа ExamTaskC содержит задачи повышенного уровня сложности
- Первая подгруппа (82 задачи) посвящена **обработке сложных наборов данных** (записей) с полями различных типов
- Вторая подгруппа (18 задач) посвящена **обработке наборов символов и строк**
- Третья подгруппа (50 задач) включает различные задачи на обработку числовых наборов данных, в том числе:
  - **Элементы с заданной разностью индексов**
  - **Делимость произведения элементов**
  - **Использование рекуррентных соотношений**
  - **Выбор чисел из набора пар**
  - **Нахождение количества пар элементов или пар с указанными свойствами**

Особенности выполнения заданий групп Exam



- В новой версии задачника PT for Exam для системы PascalABC.NET все исходные данные каждой задачи содержатся в **текстовом файле**, причем учебной программе сообщается имя этого файла.
- Выводить результаты надо на экран, используя **стандартные средства вывода** (как правило, процедуру Write/Writeln)
- Учащийся должен обеспечивать надлежащее **форматирование выходных данных** (в других группах заданий это не требуется, так как средства вывода электронного задачника выполняют форматирование автоматически). Информация о требуемых атрибутах форматирования выводится в нижней части окна задачника.

Окно задачника с задачей из группы ExamBegin



Programming Taskbook - Электронный задачник по программированию [PascalABC.NET]

МИНИМУМЫ И МАКСИМУМЫ
Задание: ExamBegin53°

Демо-запуск: Иванов Петр

Цвет (F3) Режим (F4)
Дата, время: 26/10 13:57

Новые данные (Space) | Предыдущее задание (BS) | Следующее задание (Enter) | Выход (Esc)

На вход в первой строке подается целое число N (>1), а во второй строке — массив из N различных вещественных чисел. Выполнив однократный просмотр массива, найти два его наименьших элемента и вывести эти элементы в порядке возрастания их значений (каждый элемент выводить на новой строке).

Исходные данные
Файл с исходными данными: 'pt12158.tst' Результаты надо выводить на экран.
1: '10'
'-34.86 -42.70 28.01 75.76 65.41 32.57 -45.70 91.02 -44.42 -38.95'

Пример верного решения
1: '-45.70'
'-44.42'

```
// Для вывода используйте процедуру Writeln,
// после выводимых чисел указывайте атрибуты форматирования :0:2
```

Окно задачника с задачей из группы ExamTaskC

Programming Taskbook - Электронный задачник по программированию [PascalABC.NET]

ВЫБОР ЧИСЕЛ ИЗ НАБОРА ПАР
Задание: ExamTaskC145°
Демо-запуск: Иванов Петр
Цвет (F3) Режим (F4)
Дата, время: 26/10 13:53

Новые данные (Space) | Предыдущее задание (BS) | Следующее задание (Enter) | Выход (Esc)

Дано целое число N (>1) и набор из N пар положительных целых чисел (в первой строке вводится количество пар N, в последующих строках вводятся пары чисел, причем каждая пара вводится с новой строки, а числа в паре разделяются ровно одним пробелом). Числа в парах не превосходят 1000. Необходимо выбрать из каждой пары ровно одно число так, чтобы сумма всех выбранных чисел делилась на 4 и при этом была минимально возможной. Требуется вывести найденную сумму. Если получить сумму с указанными свойствами нельзя, то в качестве ответа нужно вывести 0. При решении задачи не следует использовать полный перебор вариантов.

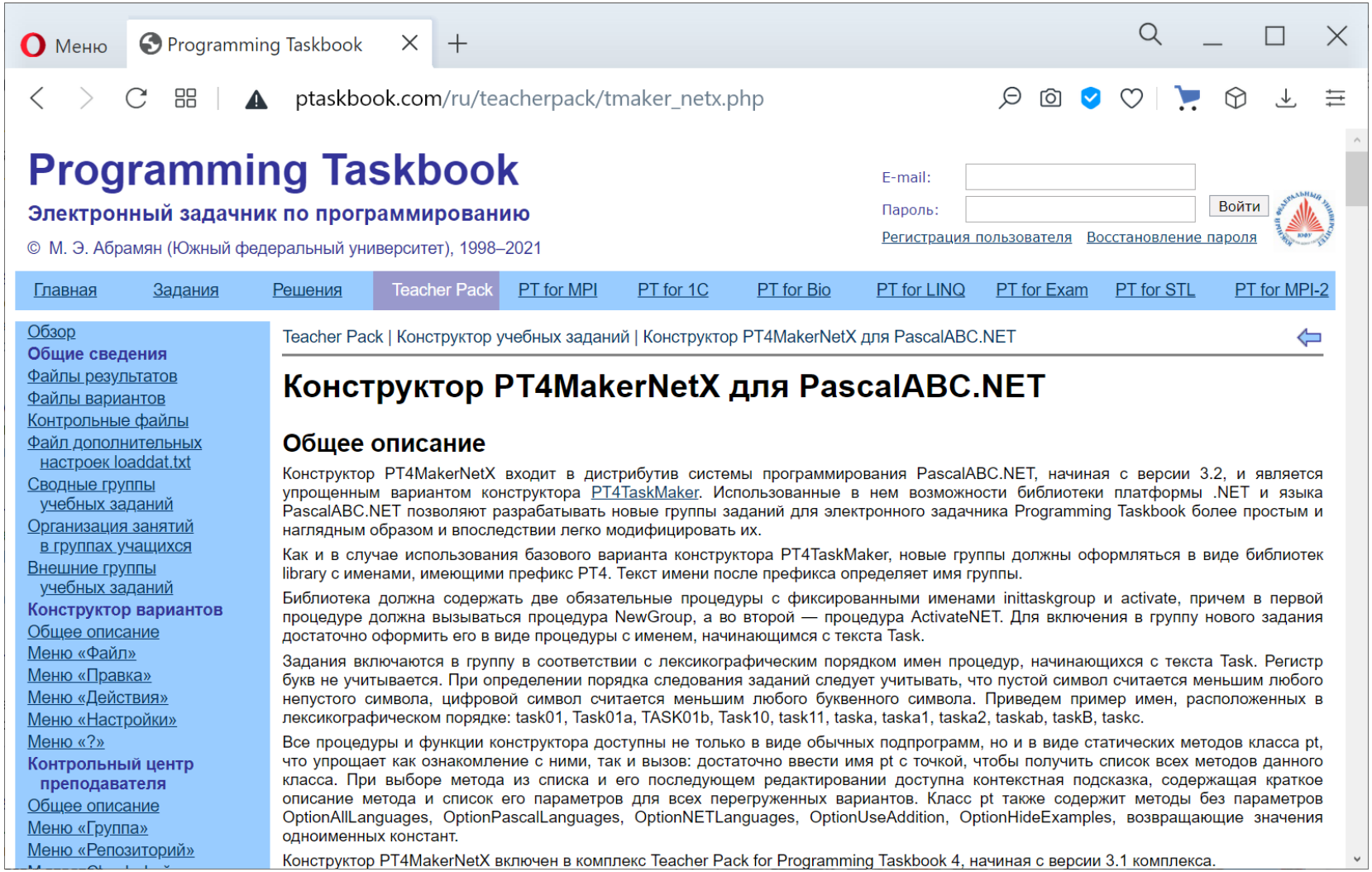
Исходные данные
Файл с исходными данными: 'pt113210.tst' Результаты надо выводить на экран.
1: '21'
'75 92'
'14 49'
'97 96'
...

Пример верного решения
1: '1104'

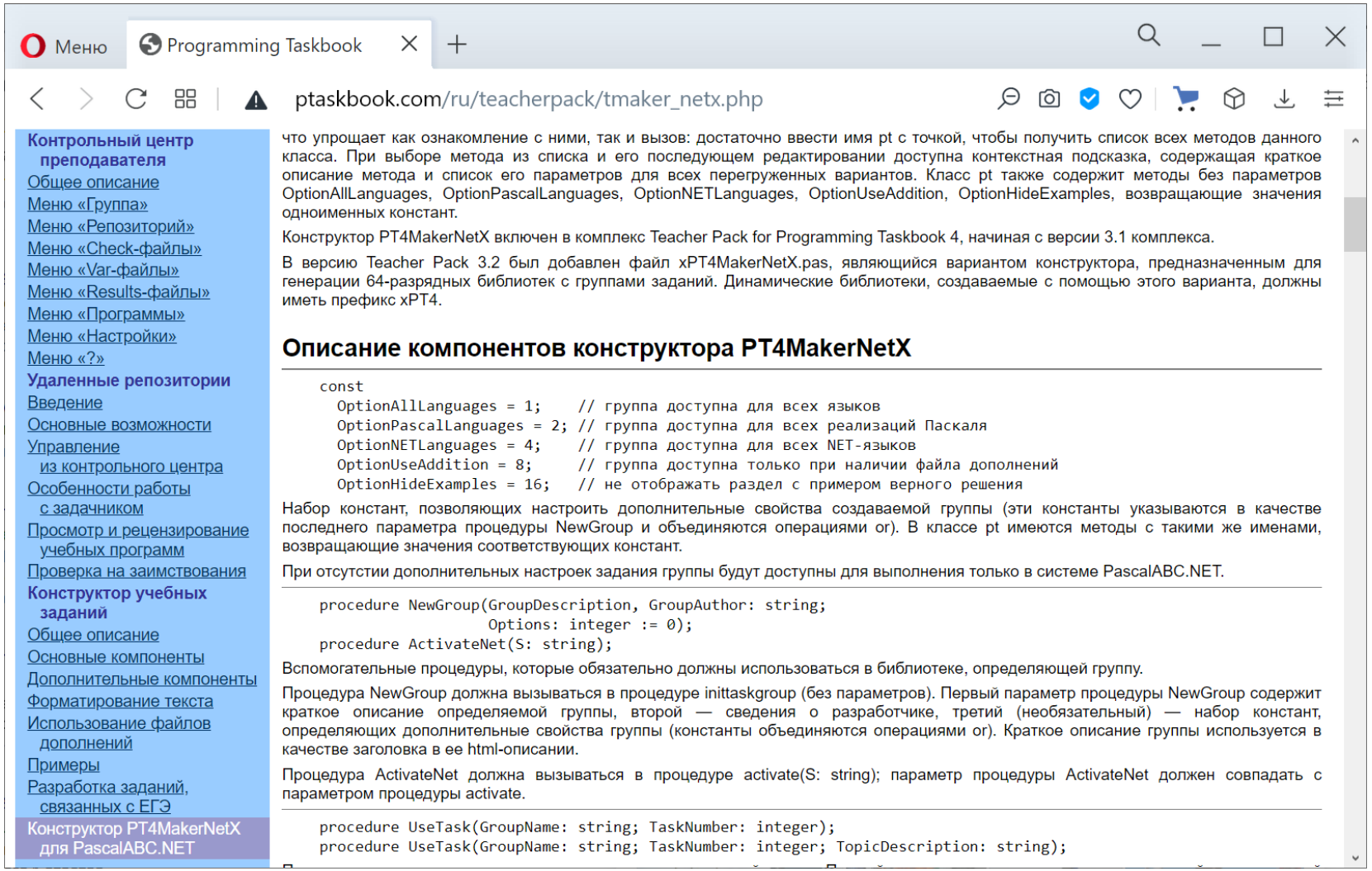
Разработка заданий с помощью конструктора PT4MakerNetX

- В систему PascalABC.NET входит **конструктор PT4MakerNetX**, позволяющий разрабатывать новые группы заданий для задачника Programming Taskbook, в том числе включающие задания на обработку файлов.
- Группа заданий оформляется в виде **динамической библиотеки** (library) с именем PT4<имя группы>, например, PT4EGE21F.pas. После компиляции мы получим файл PT4EGE21F.dll, который достаточно скопировать в рабочий каталог ученика или системный каталог задачника: Program Files (x86)\PascalABC.NET\PT4\Lib
- Описание конструктора PT4MakerNetX и пример его использования содержится на **сайте электронного задачника http://ptaskbook.com**.

Раздел сайта с описанием конструктора PT4MakerNetX (1)



Раздел сайта с описанием конструктора PT4MakerNetX (2)



что упрощает как ознакомление с ними, так и вызов: достаточно ввести имя pt с точкой, чтобы получить список всех методов данного класса. При выборе метода из списка и его последующем редактировании доступна контекстная подсказка, содержащая краткое описание метода и список его параметров для всех перегруженных вариантов. Класс pt также содержит методы без параметров OptionAllLanguages, OptionPascalLanguages, OptionNETLanguages, OptionUseAddition, OptionHideExamples, возвращающие значения одноименных констант.

Конструктор PT4MakerNetX включен в комплекс Teacher Pack for Programming Taskbook 4, начиная с версии 3.1 комплекса.

В версию Teacher Pack 3.2 был добавлен файл xPT4MakerNetX.pas, являющийся вариантом конструктора, предназначенным для генерации 64-разрядных библиотек с группами заданий. Динамические библиотеки, создаваемые с помощью этого варианта, должны иметь префикс xPT4.

Описание компонентов конструктора PT4MakerNetX

```
const
  OptionAllLanguages = 1;    // группа доступна для всех языков
  OptionPascalLanguages = 2; // группа доступна для всех реализаций Паскаля
  OptionNETLanguages = 4;    // группа доступна для всех NET-языков
  OptionUseAddition = 8;     // группа доступна только при наличии файла дополнений
  OptionHideExamples = 16;   // не отображать раздел с примером верного решения
```

Набор констант, позволяющих настроить дополнительные свойства создаваемой группы (эти константы указываются в качестве последнего параметра процедуры NewGroup и объединяются операциями or). В классе pt имеются методы с такими же именами, возвращающие значения соответствующих констант.

При отсутствии дополнительных настроек задания группы будут доступны для выполнения только в системе PascalABC.NET.

```
procedure NewGroup(GroupDescription, GroupAuthor: string;
                   Options: integer := 0);
procedure ActivateNet(S: string);
```

Вспомогательные процедуры, которые обязательно должны использоваться в библиотеке, определяющей группу.

Процедура NewGroup должна вызываться в процедуре inittaskgroup (без параметров). Первый параметр процедуры NewGroup содержит краткое описание определяемой группы, второй — сведения о разработчике, третий (необязательный) — набор констант, определяющих дополнительные свойства группы (константы объединяются операциями or). Краткое описание группы используется в качестве заголовка в ее html-описании.

Процедура ActivateNet должна вызываться в процедуре activate(S: string); параметр процедуры ActivateNet должен совпадать с параметром процедуры activate.

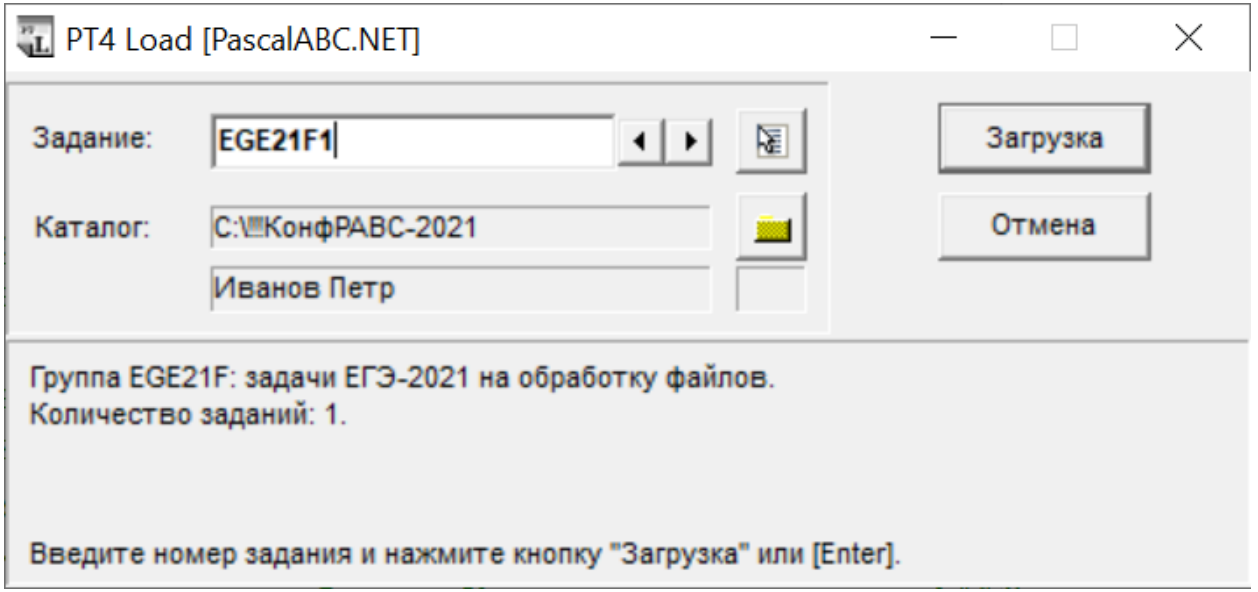
```
procedure UseTask(GroupName: string; TaskNumber: integer);
procedure UseTask(GroupName: string; TaskNumber: integer; TopicDescription: string);
```

Структура файла с группой заданий

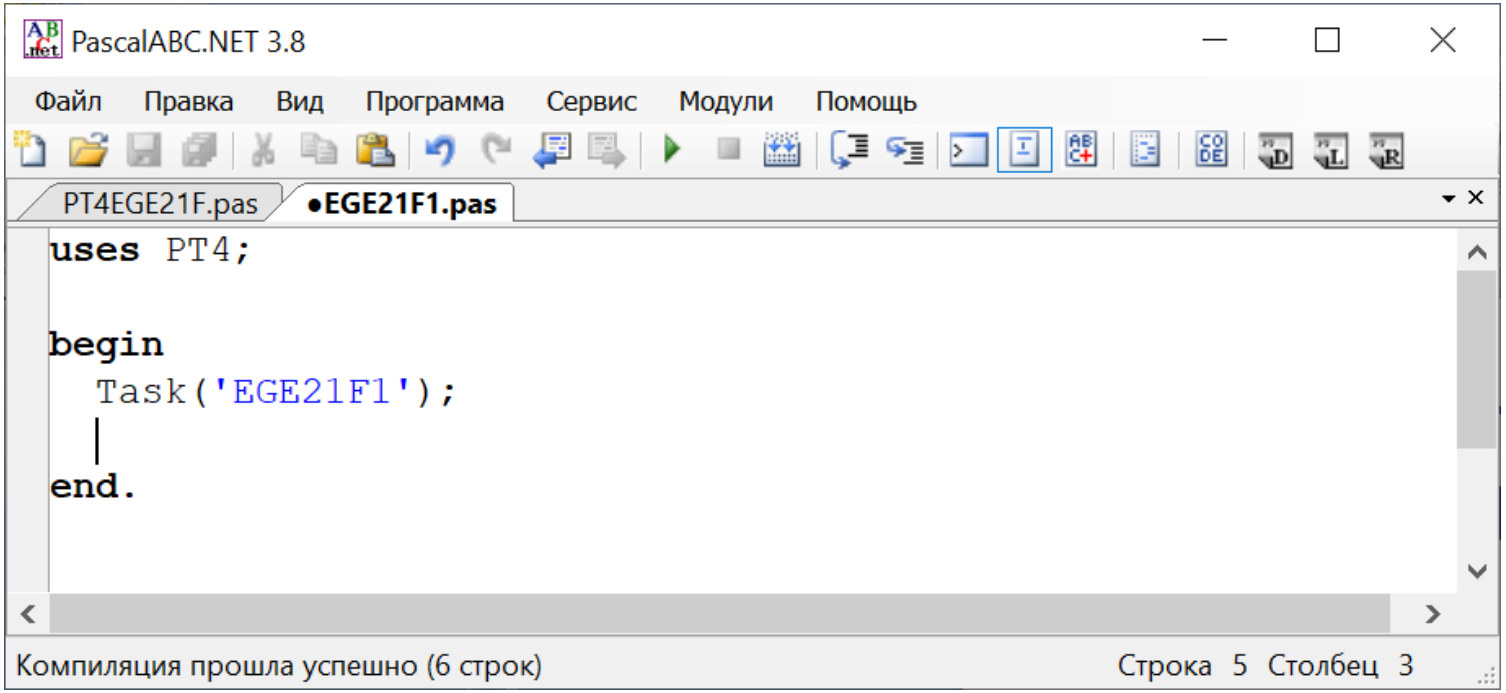
```
library PT4EGE21F; // имя библиотеки начинается с префикса PT4,
                   // после которого идет имя группы
uses PT4MakerNetX; // подключение конструктора
procedure inittaskgroup; // процедура инициализации группы должна иметь
                         // имя inittaskgroup, набранное маленькими буквами
begin
  pt.NewGroup('Задачи ЕГЭ-2021 на обработку файлов', ' М. Э. Абрамян, 2021');
end;

procedure activate(S: string); // в библиотеке должна быть описана процедура
                               // с именем activate (маленькими буквами),
                               // в которой вызывается процедура ActivateNet
begin
  pt.ActivateNet(S);
end;

procedure Task1a; // процедуры, реализующие задания
begin
end;

end.
```

Тестирование созданной группы заданий (1)

- После добавления в группу хотя бы одного задания и успешной компиляции программы (в результате будет создан dll-файл) можно создать заготовку для задания из этой группы, используя программу PT4Load (команда «Модули / Создать шаблон программы»).

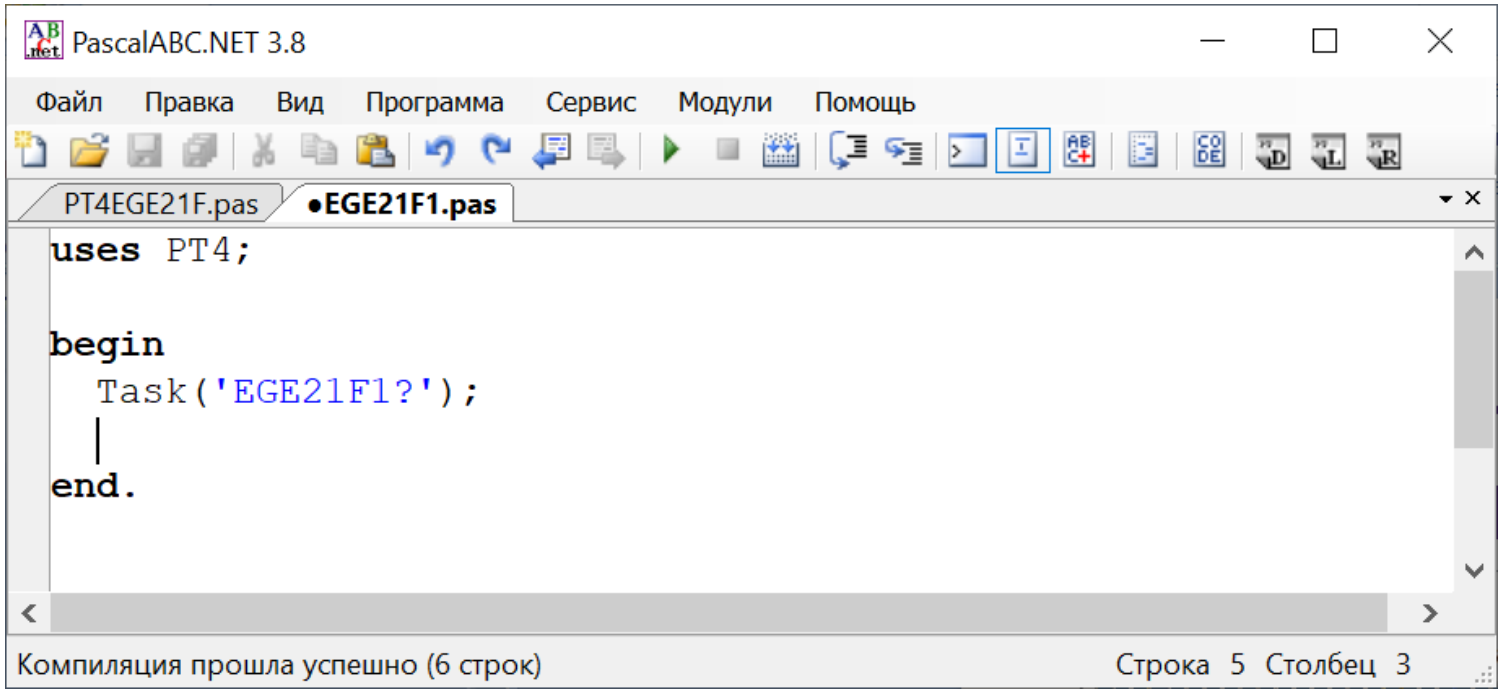


Тестирование созданной группы заданий (2)



- Запустив созданную заготовку, мы увидим окно задачника с разработанным заданием. В дальнейшем при корректировке файла с группами заданий и его перекомпиляции созданная заготовка будет запускаться автоматически.
- Для тестирования другого задания группы достаточно изменить имя задания в созданной заготовке.

Тестирование созданной группы заданий (3)



- Можно запустить задание в демонстрационном режиме (для этого достаточно добавить к имени задания символ «?»).
- Это позволит при одном запуске просмотреть несколько вариантов исходных данных и примеров правильного решения, а также сразу просмотреть все задания группы.

Пример разработки нового задания

Задание 24 № 27699

Текстовый файл состоит не более чем из 10^6 символов *L*, *D* и *R*. Определите максимальную длину цепочки вида *LDRLDRLDR*... (составленной из фрагментов *LDR*, последний фрагмент может быть неполным).

Для выполнения этого задания следует написать программу. Ниже приведён файл, который необходимо обработать с помощью данного алгоритма.

- Целесообразно разработать несколько аналогичных заданий. В данном случае можно взять указанное задание на основу, а затем привести его модификацию, в которой требуемые символы и их количество задается в том же исходном файле.
- Поскольку данная серия заданий будет содержать почти одинаковый программный код, целесообразно оформить его в виде вспомогательной процедуры.

Разработка вспомогательной процедуры (1)

```
procedure Letters(randomLetters: boolean);
begin
  var s := 'LDR';
  if randomLetters then
  begin
    s := '';
    loop Random(3, 6) do
    begin
      var c := chr(Random(ord('A'), ord('Z')));
      while Pos(c, s) > 0 do
        c := chr(Random(ord('A'), ord('Z')));
      s += c;
    end;
  end;
  var n := s.Length;
```

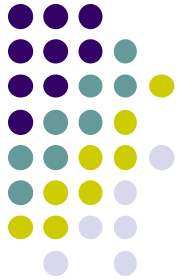
- Первый этап процедуры: определение исходного набора символов. Этот набор либо является фиксированным, либо генерируется с помощью датчика случайных чисел, причем его длина может быть различной. Набор сохраняется в строке s.

Разработка вспомогательной процедуры (2)

```
var ss := '';
loop Random(65, 74) do
  ss += s[Random(1, s.Length)];
var st := 15 * s;
loop Random(4, 6) do
begin
  var k := Random(3, 20);
  var p := Random(1, ss.Length - k + 1);
  delete(ss, p, k);
  insert(copy(st, 1, k), ss, p);
end;
```

- Второй этап процедуры: генерация строки ss для поиска. Задачник позволяет записывать в файл строки длиной до 75 символов; этого достаточно для тестирования любых алгоритмов.
- После заполнения строки ss случайными символами в нее добавляется от 4 до 6 требуемых последовательностей различной длины.

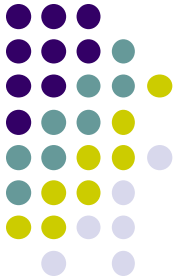
Разработка вспомогательной процедуры (3)



```
var maxlen := 0;
var len := 0;
var j := 1;
for var i := 1 to ss.Length do
begin
  if ss[i] = s[j] then
  begin
    len += 1;
    if len > maxlen then
      maxlen := len;
    j := j mod n + 1;
  end
  else if ss[i] = s[1] then
  begin
    len := 1;
    if len > maxlen then
      maxlen := len;
    j := 2;
  end
  else
  begin
    len := 0;
    j := 1;
  end;
end;
```

- Третий этап: нахождение вхождения наибольшей длины. Использован эффективный однопроходный алгоритм.

Разработка вспомогательной процедуры (4)



```
var name := pt.RandomName(8) + '.tst';
var f := OpenWrite(name);
if randomLetters then
  f.Writeln(s);
f.Write(ss);
f.Close;
pt.Data('Имя файла: ', name);
pt.DataText(name);
pt.Res('', maxlen);
```

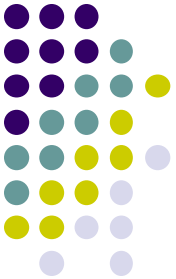
- Завершающий этап: использование средств конструктора для задания исходных и результирующих данных. Все эти средства оформлены как методы класса pt:
  - RandomName – генерация случайного имени файла,
  - Data – задание имени файла как элемента исходных данных,
  - DataText – задание самого исходного файла,
  - Res – задание требуемого результата (максимальной длины).
- Также использованы стандартные средства для работы с файлами.

Разработка процедур, связанных с заданиями (1)

```
procedure Task1a;
begin
  pt.NewTask(
  'Дано имя текстового файла, который состоит из символов L, D и R.'#13
  'Определите максимальную длину цепочки вида LDRLDRLDR\., составленной'#13
  'из фрагментов LDR (последний фрагмент может быть неполным, например, LDRLD).');
  Letters(False);
end;
```

- Каждое задание оформляется в виде процедуры, имя которой начинается с текста Task. Последующие символы имени используются для определения порядка следования заданий в группе.
- Поскольку основные действия по формированию задания уже реализованы во вспомогательной процедуре Letters, в процедуре с заданием осталось задать формулировку задания и вызвать процедуру Letters с соответствующим параметром.
- Формулировка задания задается с помощью метода pt.NewTask. В ней можно использовать управляющие последовательности (например, \. для генерации многоточия).

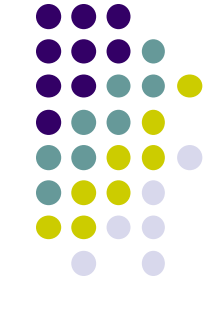
Разработка процедур, связанных с заданиями (2)

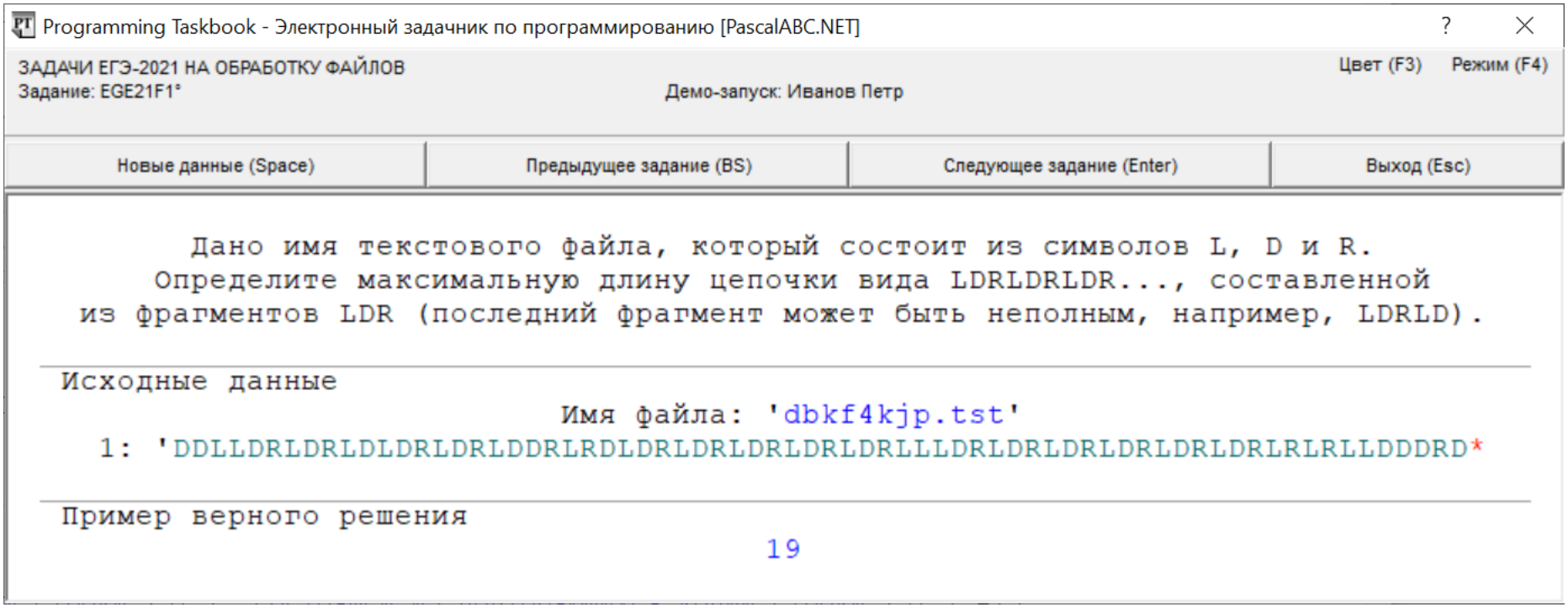


```
procedure Task1b;
begin
  pt.NewTask(
  'Дано имя текстового файла, содержащего две строки: {S}_1 и {S}_2.'#13
  'Строка {S}_1 состоит из заглавных латинских букв (все буквы в ней различны),'#13
  'строка {S}_2 состоит из букв, входящих в строку {S}_1. Определите максимальную'#13
  'длину цепочки строки {S}_2, состоящей из повторяющихся копий строки {S}_1'#13
  '(последняя копия может быть неполной). Например, для строки {S}_1~=~''ABC'''#13
  'надо анализировать цепочки вида ABCABCABC\., причем цепочки ABCA и ABCAB'#13
  'тоже считаются допустимыми.');
  Letters(True);
end;
```

- Второе, более сложное задание имеет и более длинную формулировку. В ней использованы управляющие последовательности:
  - {S} (отображает символ S курсивом),
  - _1 и _2 (отображает числа 1 и 2 в виде нижних индексов),
  - ~ (неразрывный пробел).

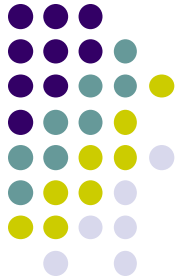
Просмотр разработанных заданий (1)





- Для просмотра заданий достаточно запустить ранее созданную заготовку.

Просмотр разработанных заданий (2)



Programming Taskbook - Электронный задачник по программированию [PascalABC.NET]

ЗАДАЧИ ЕГЭ-2021 НА ОБРАБОТКУ ФАЙЛОВ
Задание: EGE21F2*

Демо-запуск: Иванов Петр

Цвет (F3) Режим (F4)

Новые данные (Space) | Предыдущее задание (BS) | Следующее задание (Enter) | Выход (Esc)

Дано имя текстового файла, содержащего две строки: S_1 и S_2. Строка S_1 состоит из заглавных латинских букв (все буквы в ней различны), строка S_2 состоит из букв, входящих в строку S_1. Определите максимальную длину цепочки строки S_2, состоящей из повторяющихся копий строки S_1 (последняя копия может быть неполной). Например, для строки S_1 = 'ABC' надо анализировать цепочки вида ABCABCABC..., причем цепочки ABCA и ABCAB тоже считаются допустимыми.

Исходные данные

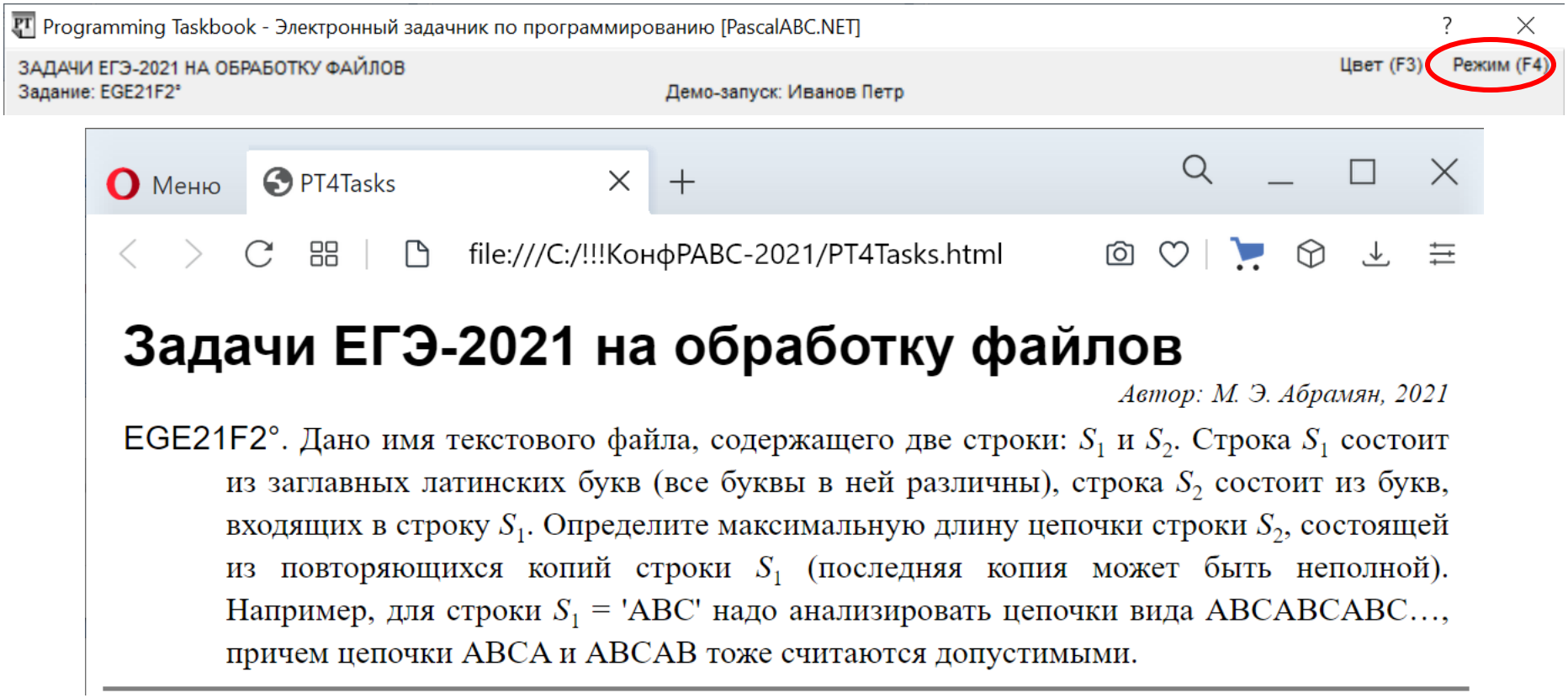
Имя файла: 'ciehymvn.tst'

1: 'AEFOWN'
'WFEOFAWAAFCAOWAAEFOWNAEFOEFOWNAEFOWNAEAEFOWNAEFFAOOWWEEOWWOOEAONWEFEAO'

Пример верного решения

10

Просмотр разработанных заданий (3)



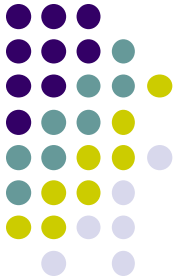
- Нажатие на метку «Режим» позволяет просмотреть формулировку задания в html-формате.

Тестирование разработанных заданий (1)

```
uses PT4;

begin
  Task('EGE21F2');
  // ОЧЕНЬ неэффективное решение!
  var f := OpenRead(ReadString);
  var s := f.ReadString;
  var ss := f.ReadString;
  s := s * (ss.Length div s.Length + 1);
  var res := 0;
  for var i := 1 to ss.Length do
    if Pos(Copy(s, 1, i), ss) > 0 then
      res := i;
  Print(res);
end.
```

- Для тестирования разработанного задания полезно решить его другим, более простым (хотя и неэффективным) способом.

Тестирование разработанных заданий (2)



Programming Taskbook - Электронный задачник по программированию [PascalABC.NET]

ЗАДАЧИ ЕГЭ-2021 НА ОБРАБОТКУ ФАЙЛОВ
Задание: EGE21F2°
Выполняет: Иванов Петр
Результаты (F2) Цвет (F3) Режим (F4)
Дата, время: 27/10 18:46

Задание выполнено!
Выход (Esc)

Ввод: 1 из 1
Вывод: 1 из 1
Тесты: 5 из 5

Дано имя текстового файла, содержащего две строки: S_1 и S_2. Строка S_1 состоит из заглавных латинских букв (все буквы в ней различны), строка S_2 состоит из букв, входящих в строку S_1. Определите максимальную длину цепочки строки S_2, состоящей из повторяющихся копий строки S_1 (последняя копия может быть неполной). Например, для строки S_1 = 'ABC' надо анализировать цепочки вида ABCABCABC..., причем цепочки ABCA и ABCAB тоже считаются допустимыми.

Исходные данные

Имя файла: 'x2ob33ha.tst'

1: 'PYHFMJ'
2: 'JMPYFFPHJPHPYHFPYHFMJPYHFMJPYHFJPYHFMJPYHFMJPYHFMJFPYHFMJPYHFMJPYHYFMM'

Полученные результаты

18