**Лабораторная работа 10 (6 час)**

**Языки программирования**

**Первый этап разработки транслятора**

**(обработка ошибок, обработка параметров,**

**ввод и проверка входных данных,**

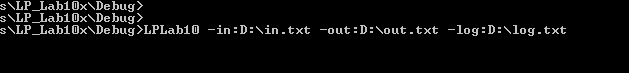
**протоколирование)**

1. Используйте материал лекции № 11.
2. Создайте проект-приложение с именем **LPLab10**.
3. Ознакомьтесь с рисунком 1, демонстрирующим схему работы приложения **LPLab10.**



Рис.1 Схема работы приложения **LPLab10**

1. Приложение **LPLab10** предназначено для вызова в консоли (командной строкой).
2. Приложение принимает параметры заданные ключами: **-in**, **-out**, **-log**.



1. Параметр **–in** является обязательным и задает полное имя файла с исходным кодом.
2. Параметр **–out** является необязательным и задает полное имя файла с объектным кодом. В том случае, если параметр **-out** не задан, то используется имя файла, образованное от имени файла с исходным кодом (**-in**) путем добавления расширения **.out**. Например, если задан параметр **-in:D:\Folder1\infile.txt** и не задан параметр **–out**, то для файла с объектным кодом используется имя **D:\Folder1\infile.txt.out**.
3. Параметр **–log** является необязательным и задает полное имя файла протокола. В том случае, если параметр **–log** не задан, то используется имя файла, образованное от имени файла с исходным кодом (**-in**) путем добавления расширения **.log**. Например, если задан параметр **-in:D:\Folder1\infile.txt** и не задан параметр **–log**, то для файла протокола используется имя **D:\Folder1\infile.txt.log**.
4. Приложение **LPLab10** посимвольносчитывает файл с исходным кодом в оперативную память. При считывании осуществляет проверку символов на допустимость. В процессе обработки входных параметров или считывании файла с исходным кодом могут возникать ошибки, которые фиксируются в протоколе (если он к этому времени уже создан) или выводятся на консоль (если протокол не создан).
5. Разработку приложения следует выполнять в следующей последовательности:

- функции для обработки ошибок;

- функции для обработки входных параметров;

- функции для ввода файла исходных кодов;

- функции для работы с протоколом.

1. **Обратите внимание**: разрабатываемое в рамках данной лабораторной работы приложение **LPLab10**, ***не должно формировать файл с объектным кодом.***
2. Пространства имен и имена файлов с исходным кодом для каждого набора функций сведены в следующей таблице 1.

Таблица 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Набор функций** | **Пространство**  **имен (namespace)** | **Заголовочный**  **файл (\*.h)** | **Реализация**  **(\*.cpp)** |
| обработка ошибок | Error | Error.h | Error.cpp |
| обработка параметров | Parm | Parm.h | Parm.cpp |
| ввод исходного кода | In | In.h | In.cpp |
| работа с протоколом | Log | Log.h | Log.cpp |

**Обработка ошибок**

1. Ознакомьтесь с содержимым файла **Error.h** (рис.4).
2. Ознакомьтесь с содержимым файла **Error.cpp** (рис.5, реализация функций **geterror** и **geterrorin** намерено скрыта).
3. На рис.2 приведена структура приложения **LPLab10**.

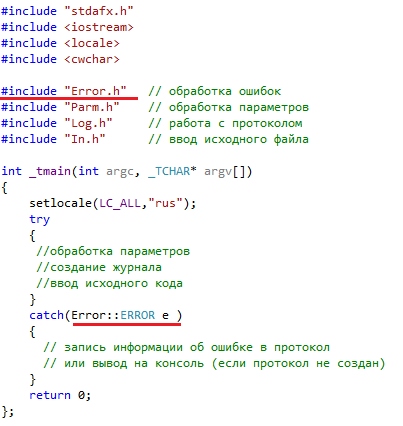


Рис. 2. Структура приложения

1. Разработайте функции **geterror** и **geterrorin**. Описание их приведено в таблице 2. Указание:

Таблица 2

|  |  |
| --- | --- |
| **Наименование**  **функции** | **Назначение** |
| geterror | Используется в макросе **ERROR\_THROW**.  **Параметры:** **id** - код ошибки (int).  **Выполняет**: проверяет допустимый диапазон **id**; извлекает данные из таблицы ошибок и заносит данные в возвращаемую структуру **ERROR**. Если значение параметра **id** выходит за переделы допустимого диапазона (0 < id < **ERROR\_MAX\_ENTRY**), то формируется содержимое структуры **ERROR** соответствующее ошибки с кодом 0.  **Возврат:** заполненная структура **ERROR**. |
| geterrorin | Используется в макросе **ERROR\_THROW\_IN**.  **Параметры:** **id** - код ошибки (int), **line** – номер строки (int, по умолчанию **-1**), **col** – полиция в строке (int, по умолчанию **-1**).  **Выполняет**: проверяет допустимый диапазон **id**; извлекает данные из таблицы ошибок и заносит данные в возвращаемую структуру **ERROR**. Если значение параметра **id** выходит за переделы допустимого диапазона (0 < id < **ERROR\_MAX\_ENTRY**), то формируется содержимое структуры **ERROR** соответствующее ошибки с кодом 0.  **Возврат:** заполненная структура **ERROR.** |

1. На рис. 3 приведен программный код, тестирующий функции **geterror** и **geterrorin.**



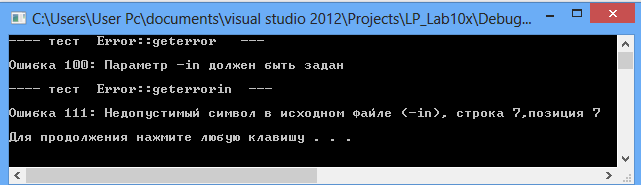


Рис. 3. Тест функций **geterror** и **geterrorin**

1. Выполните тест и убедитесь в работоспособности функций **geterror** и **geterrorin** и макросов **ERROR\_THROW** и **ERROR\_THROW\_IN**.

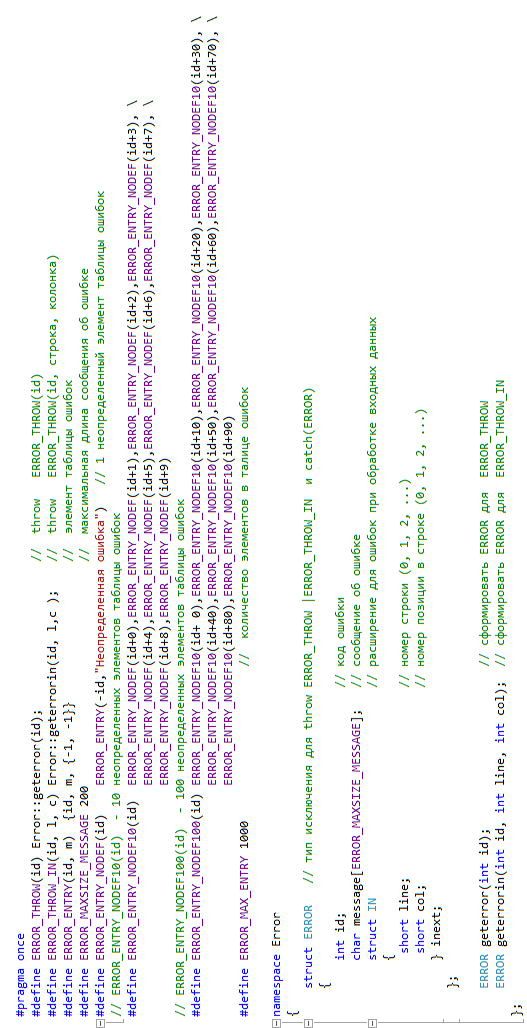


Рис.4. Содержимое файла Error.h

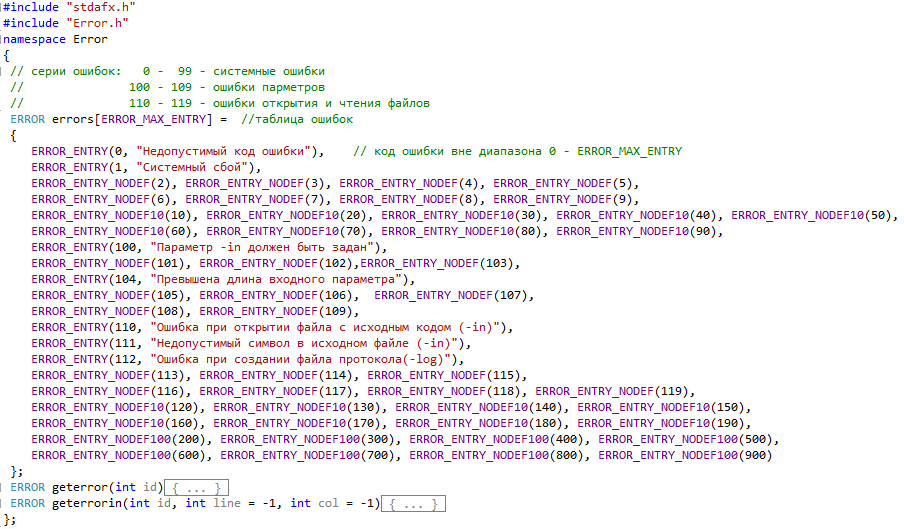


Рис.5. Содержимое файла Error.cpp

**Обработка входных параметров**

1. Ознакомьтесь с содержимым файла **Parm.h** (рис.6).

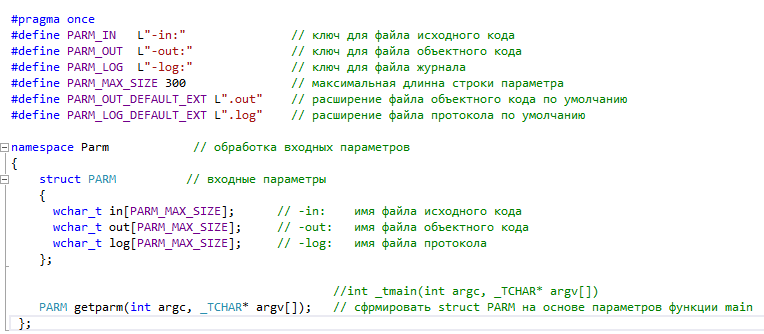
****

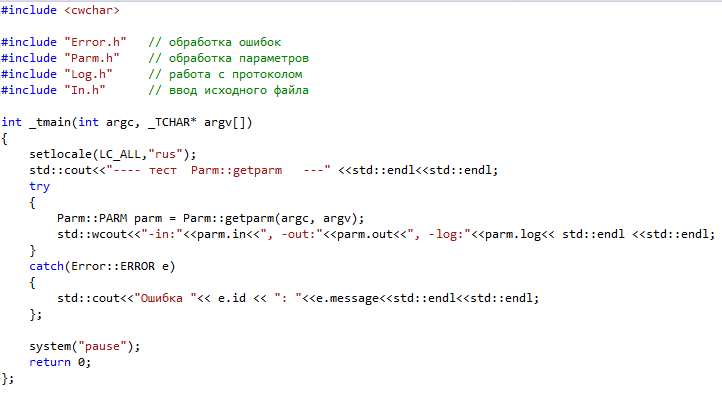
Рис.6. Содержимое файла Parm.h

1. Разработайте функцию **getparm** (табл.3). Указание: используйте функции **wcscpy\_s**, **wcsncat\_s**, **wcslen**, **wcsstr**, **wcslen** стандартной библиотеки.

Таблица 3

|  |  |
| --- | --- |
| **Наименование**  **функции** | **Назначение** |
| getparm | Используется для записи значений входных параметров **(-in:**, **-out:**, **-log:**) в структуру **PARM**.  **Параметры:** **argc** – количество параметров (int), **argv** – массив указателей на строки (\_**TCHAR\*** -указатель строку **wchar\_t**)  **Выполняет**: проверяет наличие параметра **–in:**; если параметр не задан генерируется исключение (**ERROR\_THROW**) с кодом ошибки **100;** если не задано значения **-out:** и **-log**, то формирует значения по умолчанию (см п.5-8); проверяет длину строки каждого входного параметра; если длина строки превышает значение **PARM\_MAX\_SIZE** (рис.6), то генерируется исключение (**ERROR\_THROW**) с кодом ошибки **104;**  **Возврат:** заполненная структура **PARM**. |

1. На рис. 7 приведен программный код, тестирующий функцию **getparm.**



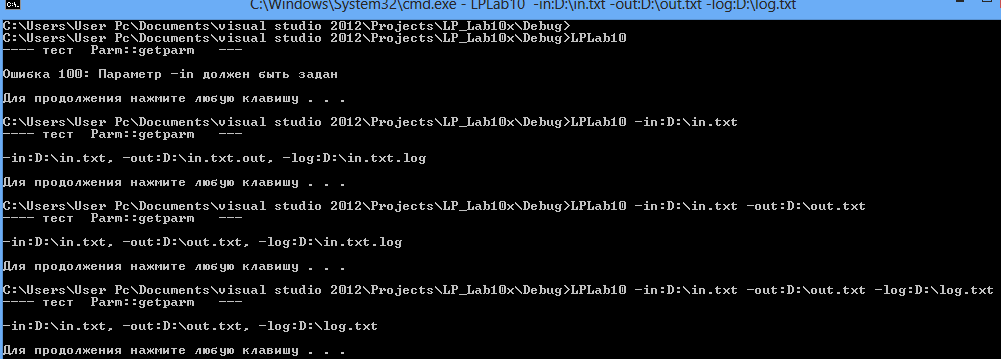


Рис. 7. Тест функции **getparm**

1. Выполните тест и убедитесь в работоспособности функций **getparm** (табл.3).

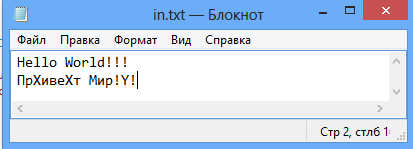
**Ввод файла исходных кодов**

1. Ознакомьтесь с содержимым файла **In.h** (рис.7).
2. Разработайте функцию **getin**. Указание: используйте потоковый ввод **ifstream** для посимвольного ввода данных.

Таблица 4

|  |  |
| --- | --- |
| **Наименование**  **функции** | **Назначение** |
| getin | Используется для ввода и проверки информации из файла с исходными кодами.  **Параметры:** infile– имя входного файла (**wchar\_t**\*)  **Выполняет**: посимвольно вводит данные из файла, заданного параметром; проверяет каждый символ на соответствие таблице проверки; подсчитывает и записывает в структуру **IN** количество введенных строк и символов, а также пропущенных символов; записывает в структуру **IN** таблицу проверки, символ может быть введен (в таблице **IN:T**), пропущен (**IN:I**), заменен (в таблице значение от **0** до **255**); если в таблице проверки символу соответствует значение **IN:F**, то генерируется исключение (**ERRROR\_THROW\_IN,** код ошибки **111**), которое фиксирует в структуре **ERROR** номер строки (отсчет от 0) и номер позиции в строке (отсчет от 0), в котором обнаружен запрещенный символ; если возникает ошибка при открытии файла выходного потока, генерируется исключение (**ERRROR\_THROW,** код ошибки 110).  **Возврат:** заполненная структура **IN**. |

1. На рис. 8 и 9 приведен исходный файл и программный код, тестирующий функцию **getin.** Проверочная таблица, допускает ввод букв входящих в выражения **Hello World!** **Привет Мир** и символ \n (конец строки), игнорирует английскую букву **X** и символ с кодом **0x0d**, а также заменяет английскую букву **Y** на **!**.





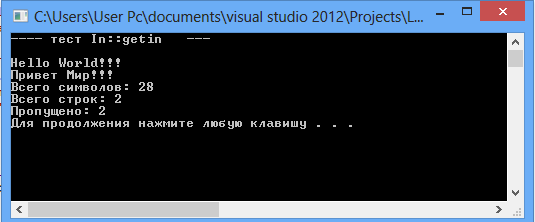
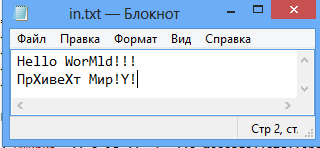
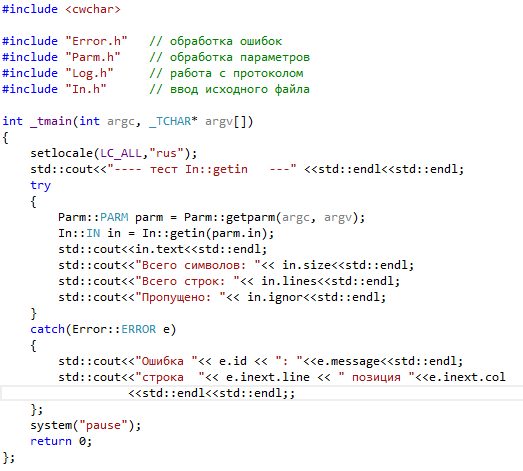


Рис. 8. Тест функции **getin**





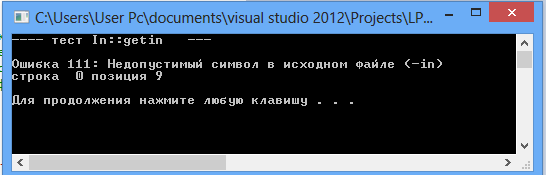


Рис. 9. Тест функции **getin**

1. Выполните тест (рис.8,9) и убедитесь в работоспособности функций **getin** (табл.4).
2. Сформируйте таблицу таким образом, чтобы допускался ввод только букв входящих в вашу фамилию и имя на русском и английском языках, а также цифр входящих в год вашего рождения. Кроме того, буква A (русская буква) должна заменяться на разрешенный символ – (минус), буква X (английская буква должна игнорироваться).
3. Протестируйте функцию **getin** (табл.4) на собственных данных.

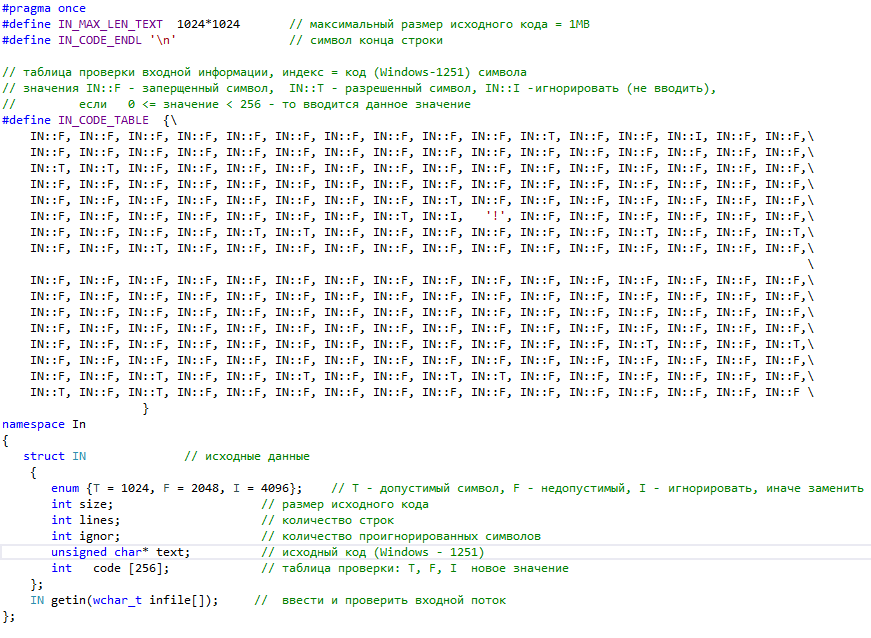


Рис.7. Содержимое файла In.h

**Работа с протоколом**

1. Ознакомьтесь с содержимым файла **Log.h** (рис.8).

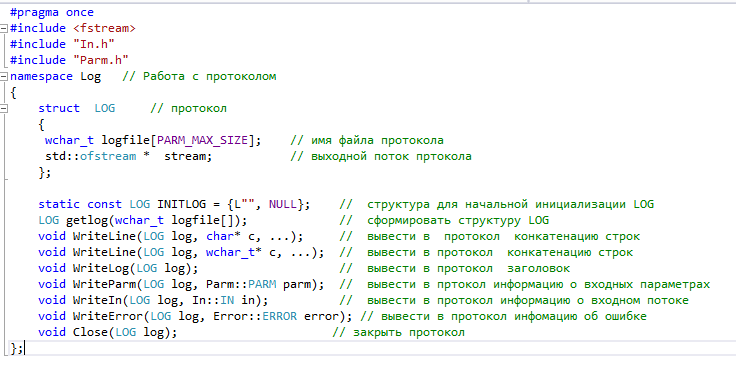


Рис.8. Содержимое файл Log.h

1. Разработайте функции, описанные в табл.5.

Таблица 5

|  |  |
| --- | --- |
| **Наименование**  **функции** | **Назначение** |
| getlog | Используется для создания и открытия потокового вывода протокола.  **Параметры:** logfile– имя входного файла (**wchar\_t**\*)  **Выполняет**: открывает (создает) выходной поток; если поток не создался, генерируется исключение (**ERRROR\_THROW,** код ошибки **112**); записывает данные в структуру **LOG**.  **Возврат:** заполненная структура **LOG**.  **Указание**: примените потоковый вывод **ofsream** |
| WriteLine  (две функции) | Используется для вывода одной строки в протокол  **Параметры:** структура **LOG**,переменное число параметров типа **char\***, последний параметр должен быть пустой строкой (“”).  **Параметры:** структура **LOG**,переменное число параметров типа **wchar\_t\***, последний параметр должен быть пустой строкой (L“”).  **Выполняет**: осуществляет конкатенацию всех строк заданных параметрами, формирует строку и выводит ее в протокол.  **Возврат:** функция ничего не возвращает  **Указание:** для преобразования строки **wchar\_t\*** в строку **char\*** примените функцию **wstombs** |
| WriteLоg | Используется для вывода заголовка протокола  **Параметры:** структура **LOG.**  **Выполняет**: выводит строку заголовка в протокол (образец в тесте).  **Возврат:** функция ничего не возвращает.  **Указание:** для получения текущей даты и времени в формате строки используйте функции **time**, **localtime\_s** и **strftime**. |
| WriteParm | Используется для вывода в протокол информации о входных параметрах  **Параметры:** структура **LOG** и структура **PARM**.  **Выполняет**: выводит в протокол информацию о параметрах (образец в тесте).  **Возврат:** функция ничего не возвращает. |
| WriteIn | Используется для вывода в протокол информации о входных данных (исходный код)  **Параметры:** структура **LOG** и структура **IN**.  **Выполняет**: выводит в протокол информацию о входных данных (образец в тесте).  **Возврат:** функция ничего не возвращает. |
| WriteError | Используется для вывода в протокол или на консоль информации об ошибке.  **Параметры:** структура **LOG** и структура **IN**.  **Выполняет**: выводит в протокол информацию об ошибке; если протокол не открыт, выводит информацию на консоль (образец в тесте)  **Возврат:** функция ничего не возвращает. |
| Close | Используется для закрытия выходного потока протокола.  **Параметры:** структура **LOG**.  **Выполняет**: закрывает выходной поток.  **Возврат:** функция ничего не возвращает. |

1. На рис. 9, 10, 11, 12 приведен исходный файл и программный код, тестирующий функции, описанные в табл. 5.

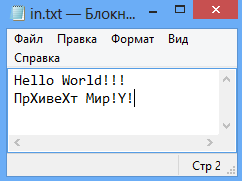


Рис.9. Исходные данные

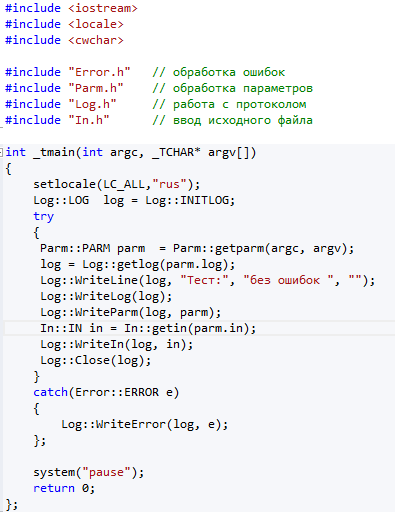


Рис. 10. Тест функций из табл. 5

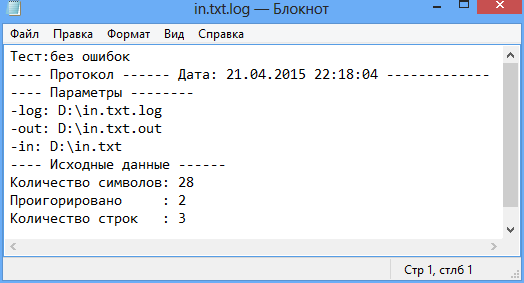


Рис. 11. Протокол без ошибок

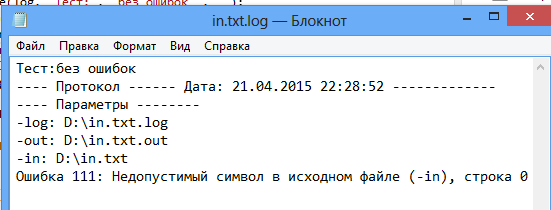


Рис. 12. Протокол ошибками