Учреждение образования

«Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 «Программное обеспечение информационных

Технологий»

ОТЧЕТ

по учебной ознакомительной практике

Исполнитель

студент \_\_ курса \_\_\_\_\_\_\_группы \_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата) (Ф.И.О.)

Руководитель практики

от университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, уч. звание) (подпись, дата) (Ф.И.О.)

Отчет защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Минск 2022 г.

**Содержание**

[Введение 3](#_Toc107451028)

[1. Выполнение лабораторной работы №15 4](#_Toc107451029)

[1.1. Цель работы 4](#_Toc107451030)

[1.2. Конструирование приложения 4](#_Toc107451031)

[1.3. Тестирование приложения 6](#_Toc107451032)

[1.4. Вывод 8](#_Toc107451033)

[2. Программа IT Warm-up 9](#_Toc107451034)

[2.1. Цели на курс 9](#_Toc107451035)

[2.2. Лекции, общие задания, курс по С# 9](#_Toc107451036)

[2.3. Плюсы и минусы данной программы 11](#_Toc107451037)

[2.4. Вывод 11](#_Toc107451038)

[Приложение 12](#_Toc107451039)

[Приложение А 12](#_Toc107451040)

[Приложение Б 13](#_Toc107451041)

[Приложение С 15](#_Toc107451042)

[Приложение Д 17](#_Toc107451043)

[Приложение Е 19](#_Toc107451044)

# Введение

На период практики, которая проходила с 20.06 по 01.07, нам предстояло выполнить ряд определенных задач, таких как:

* реализовать лабораторную работу №15 по КПО;
* пройти курс от программы IT warm-up от EPAM System;

В курс от программы IT warm-up от EPAM System входят определенные задания, такие как:

* Прослушать базовые лекции;
* Прослушать лекции по специальностям;
* Выполнить домашнее задание;
* пройти курс по специальностям, которые были предложены нам в рамках данного курса;

# 1. Выполнение лабораторной работы №15

## 1.1. Цель работы

Цель работы — разработать проект-приложение, предназначенное для вызова в консоли (из командной строки разработчика).

Приложение должно принимать входной файл в кодировке Windows 1251. При посимвольном считывании осуществляется проверка символов на допустимость: каждый символ проверяется на соответствие таблице проверки. В процессе обработки входных параметров или считывании файла с исходным кодом могут возникать ошибки, которые фиксируются в протоколе. Приложение принимает на вход параметры, заданные ключами: -in, -out, -log. Приложение является многофайловым проектом и обеспечивает выполнение следующих функций:

− обработки и анализ входных параметров;

− ввод файла исходных кодов;

− проверка входных данных на допустимость;

− обработка ошибок;

− обеспечение работы с протоколом.

Для проверки работоспособности приложения в целом и функций, входящих в него, необходимо выполнить тестирование.

## 1.2. Конструирование приложения

Приложение будет разбито на несколько функций, которые были рассмотрены в [подпункте 1.1](#_1.1._Цель_работы).

Прототипы функция и шаблоны структур описаны в заголовочных файлах, а реализация в соответствующих cpp-файлах.

Первый этап — это разработка функций обработки ошибок. Тут будет 2 функции geterror() и geterrorin().

Функция geterror() отвечает за то, чтобы ловить ошибку при выполнения кода. Ее тип определяется структурой ERROR, в которой хранятся такие поля, как message — сообщение об ошибки, id — уникальный номер ошибки, структура inext — строка и элемент, где есть ошибка. Реализация данной функции показана в [приложении А](#_Приложение_А).

Функция geterrorin() — функция, которая обрабатывает ошибки при чтении файла с исходным кодом. Например, если в файле будет встречен запрещенный символ, данная функция заполнит структуру типа ERROR такими значениями, как сообщение об ошибке, id ошибки, строка и номер символа, где встретилась ошибка.

Но напрямую эти функции будут вызываться с помощью макросов ERROR\_THROW и ERROR\_THROW\_IN. Эти макросы просто заменяют вызов предыдущих функций.

Чтобы удобно было связывать номер ошибки с ее объяснением в виде текста, была создана таблица ошибок, которая хранит в себе все возможные ошибки. Реализация таблицы в [приложении А](#_Приложение_А).

Второй этап реализации данного приложения — реализовать функции обработки входных параметров. Реализация шаблонов структур, прототипов функций и их реализации рассмотрены в [приложении Б](#_Приложение_Б).

Функция getparm используется для записи значений входных параметров (-in:, -out:, -log:) в структуру PARM.

Параметры: argc – количество параметров (int, >=1), argv – массив указателей на нуль-терминальные строки со значениями параметров, (\_TCHAR\* – указатель на 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; проверяет длину строки каждого входного параметра; если длина строки превышает значение PARM\_MAX\_SIZE(макрос равный значению 300), то генерируется исключение (ERROR\_THROW) с кодом ошибки 104; Возврат: заполненная структура PARM. Проверка на соответствующие данные осуществляется с помощью регулярных выражений.

Следующий этап — это проверка файла с исходным кодом. За это отвечает функция getin. Реализация всей конструкции будет представлена в [приложение С](#_Приложение_С).

Функция 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 (обозначен в таблице IN:T), пропущен (IN:I), заменен на другой символ (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.

И последние функции, которые нужно реализовать за данную практику, — функция записи отчета об выполненной работе в log-файл. Реализация этих функций в [приложение Д](#_Приложение_Д).

**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.

**WriteLine** 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.

Выполняет: осуществляет конкатенацию всех строк, заданных параметрами, формирует строку и выводит ее в протокол.

Возврат: функция ничего не возвращает.

**WriteLоg** используется для вывода заголовка протокола.

Параметры: структура LOG.

Выполняет: выводит строку заголовка в протокол.

Возврат: функция ничего не возвращает.

**WriteParm** используется для вывода в протокол информации о входных параметрах.

Параметры: структура LOG и структура PARM.

Выполняет: выводит в протокол информацию о параметрах.

Возврат: функция ничего не возвращает.

**WriteIn** используется для вывода в протокол информации о входных данных.

Параметры: структура LOG и структура IN.

Выполняет: выводит в протокол информацию о входных данных.

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.

Возврат: функция ничего не возвращает.

**Close** используется для закрытия выходного потока протокола.

Параметры: структура LOG.

Выполняет: закрывает выходной поток.

Возврат: функция ничего не возвращает.

Последний этап конструирования данного приложения — в функции main последовательно вызвать все функции для их совместного использования. Содержимое функции main в [приложении Е](#_Приложение_Е).

## 1.3. Тестирование приложения

На время тестирования был создан файл in.txt, содержимое которого показано на рисунке 1.1. В файле с исходным кодом должны быть символы русского и английского алфавитов, символ восклицательного знака и символ пробела. Также английская буква «Y» заменяется на знак «!». А если встречаются другие символы, то будет ошибка.

В командную строку разработчика мы вызовем приложение с входными параметром “-in:in.txt”, что показано на рисунке 1.2, также тут показан результат выполнения программы. Также заметим, что log файл создался и в нем есть информация(рисунок 1.3).

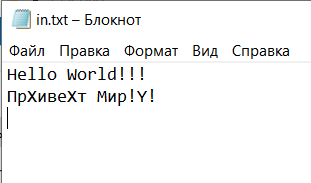


Рисунок 1.1.



Рисунок 1.2.

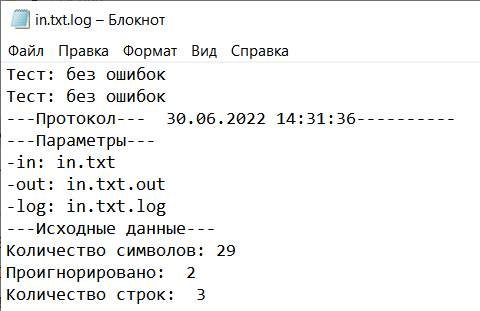


Рисунок 1.3.

Следующий этап тестирования — проверка на то, если будет в файле исходного кода запрещенный символ «З». log-файл на рисунке 1.4.



Рисунок 1.4.

Также проверим, если файла с исходным кодом не будет. Вызовем файл «-in:test.txt»(рисунок 1.5).





Рисунок 1.5.

Видим, что программа работает полностью корректно. Запись из log-файла на рисунке на 1.6.



Рисунок 1.6.

## 1.4. Вывод

В этом разделе был разбор и реализация лабораторной работы №15 по КПО. Каждая функция данного приложения была разобрана, а также данной приложение было полностью протестировано. Тестирование показало, что приложение полностью работоспособное.

# 2. Программа IT Warm-up

## 2.1. Цели на курс

За данный курс я должен познакомиться с компанией EPAM изучить основы, которые помогут войти в IT. Также программа warm-up предлагает выбрать один из нескольких языков для изучения.

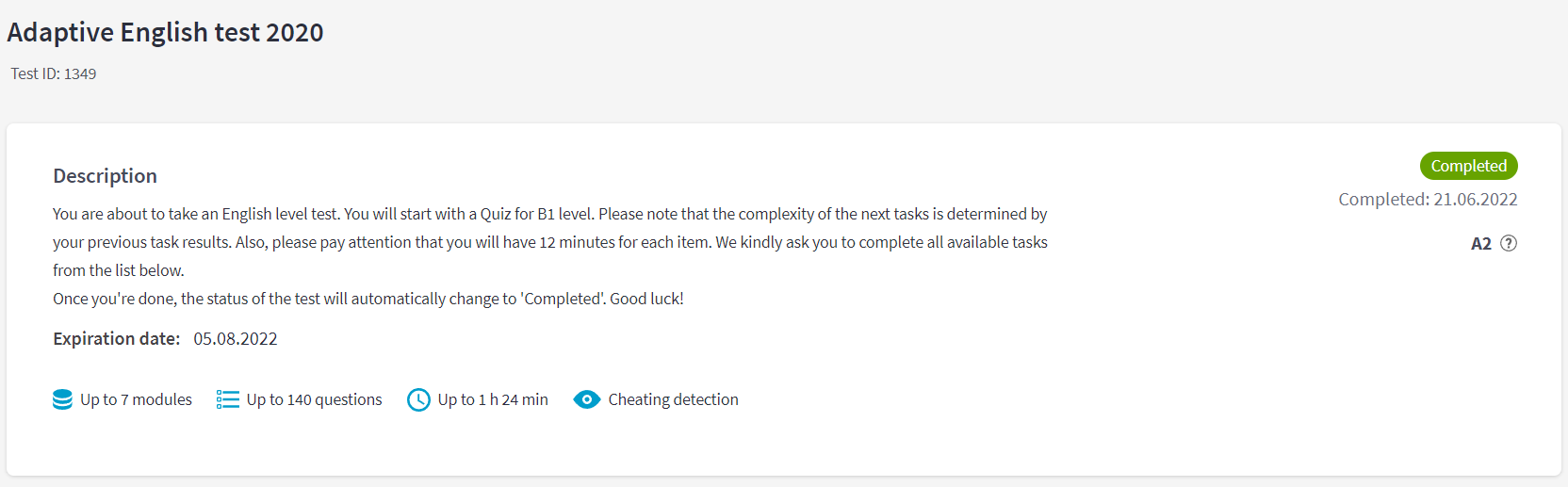
Данная программа также предлагает изучить такие технологии, как система контроля версий(GIT), С#, Java, Frontend technologies.

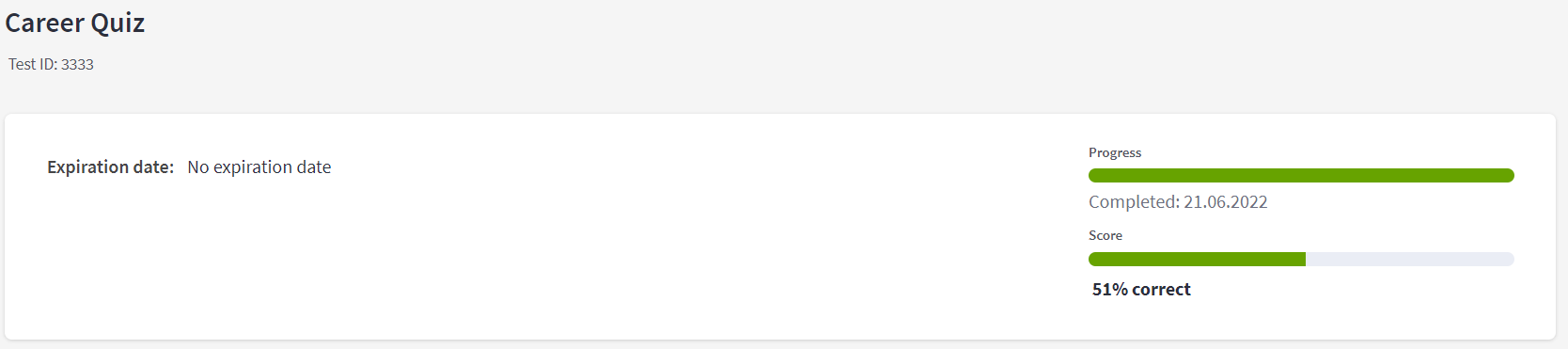
## 2.2. Лекции, общие задания, курс по С#

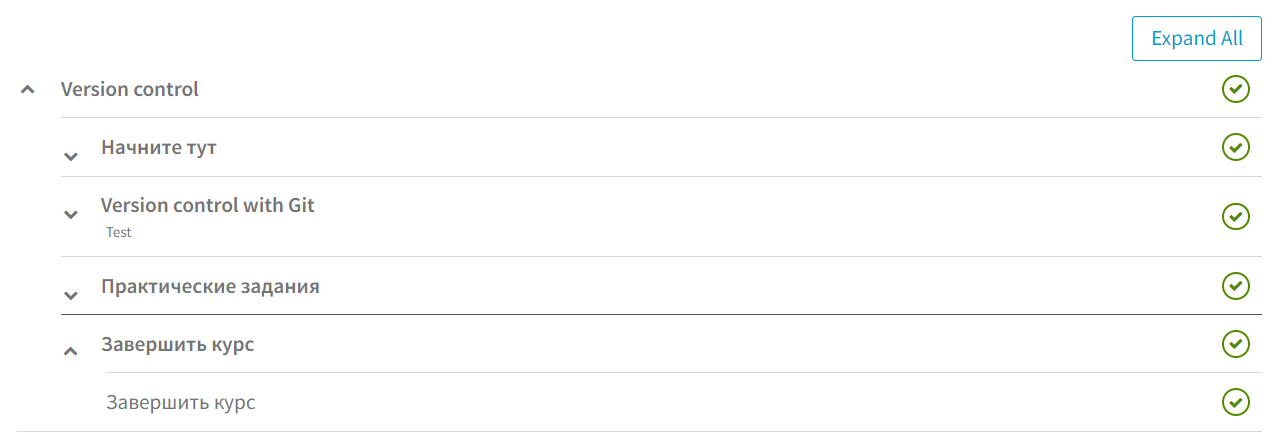
EPAM в течении практики проводил онлайн лекции о компании, различных языках программирования:

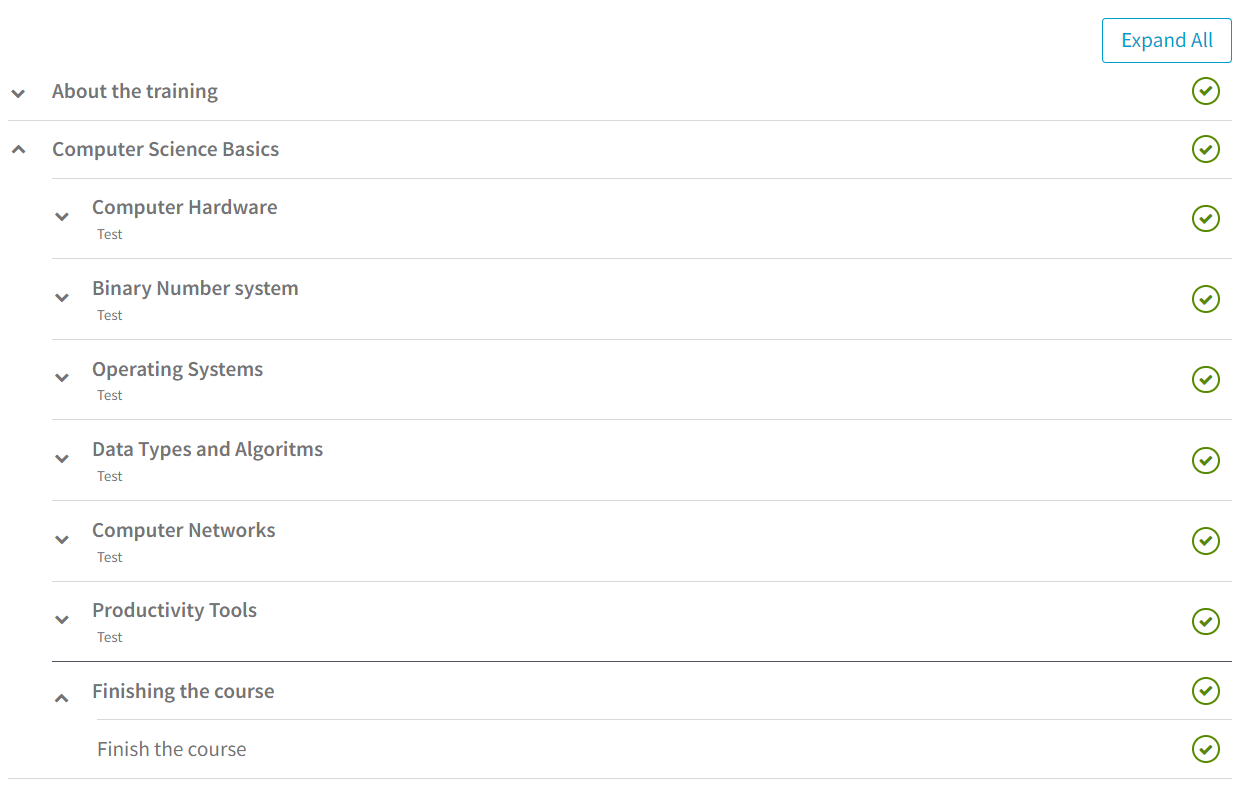
* IT Warm-Up: Introduction, EPAM & training.by portal
* IT Warm-Up: Learn.epam.com & EPAM education instuments
* IT Warm-Up: Autocode & Java
* IT Warm-Up: professions and EPAM education instruments

Также присутствовали домашние задания, представлены на рисунке 2.1.









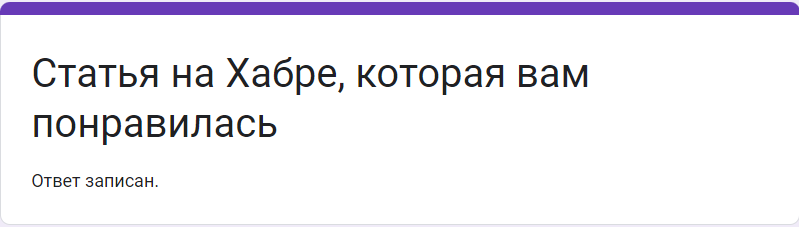


Рисунок 2.1.

EPAM предлагал нам выбрать для изучения один из языков. Мой выбор — С#.

Я выбрал данный язык, так как для меня он кажется более полезным для будущих разработок, например, курсовая работа на 4 семестре.

Для прохождения курса было необходимо создать аккаунт на GitLab.com, зарегистрировался на autocode-next.lab.epam.com со своим GitLab аккаунтом, зайти в группу .NET Development по ссылке.

Я выполнил несколько заданий из данного курса, количество баллов за каждое из ник показано на рисунке 2.2.

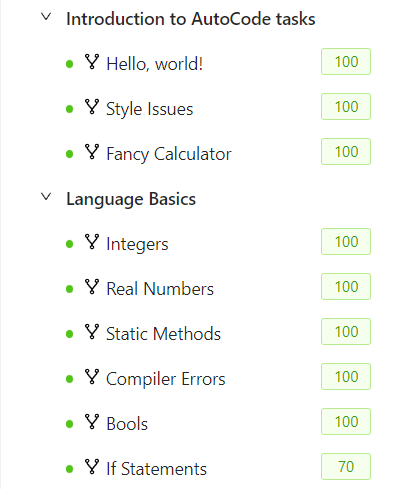






Рисунок 2.2.

## 2.3. Плюсы и минусы данной программы

|  |  |
| --- | --- |
| **Плюсы** | **Минусы** |
| Довольно-таки интересный курс для новичков. Много информации, которая пригодится для будущей работы | Слишком мало времени для изучения этого курса |
| Возможность выбора языка программирования для изучения за курс | Весь курс на английском, что усложняет его изучение |
| Много лекций по разным IT темам | Практические задания абсолютно бесполезные, так как нужно переписывать код из документаций от ЕПАМ |

## 2.4. Вывод

По итогу курс не был пройден полностью из-за недостатка времени. Практика длится 2 недели, а курс рассчитан на более.

Я бы поставил данного курсу оценку 3 из 10.

Курс рассчитан на новичков, большое количество однотипных заданий.

# Приложение

# Приложение А

|  |
| --- |
| #include "stdafx.h"  #include "Error.h"  namespace Error  {  // серии ошибок 0 - 99 - системные ошибки  // 100 - 109 - ошибки параметров  // 110 - 119 - ошибки открытия и чтения файлов  ERROR errors[ERROR\_MAX\_ENTRY] =  {  ERROR\_ENTRY(0, "Недопустимый код ошибки"),  ERROR\_ENTRY(1, "Системный сбой"),  ERROR\_ENTRY\_NODEF(2), ERROR\_ENTRY\_NODEF(3), ERROR\_ENTRY\_NODEF(4), ERROR\_ENTRY\_NODEF(5),  ERROR\_ENTRY\_NODEF(6), ERROR\_ENTRY\_NODEF(7), ERROR\_ENTRY\_NODEF(8), ERROR\_ENTRY\_NODEF(9),  ERROR\_ENTRY\_NODEF10(10),ERROR\_ENTRY\_NODEF10(20),ERROR\_ENTRY\_NODEF10(30),ERROR\_ENTRY\_NODEF10(40),  ERROR\_ENTRY\_NODEF10(50),ERROR\_ENTRY\_NODEF10(60),ERROR\_ENTRY\_NODEF10(70),ERROR\_ENTRY\_NODEF10(80),ERROR\_ENTRY\_NODEF10(90),  ERROR\_ENTRY(100,"Параметр -in должен быть задан"),  ERROR\_ENTRY\_NODEF(101),ERROR\_ENTRY\_NODEF(102),ERROR\_ENTRY\_NODEF(103),  ERROR\_ENTRY(104,"Превышена длина входного параметра"),  ERROR\_ENTRY\_NODEF(105),ERROR\_ENTRY\_NODEF(106),ERROR\_ENTRY\_NODEF(107),ERROR\_ENTRY\_NODEF(108),ERROR\_ENTRY\_NODEF(109),  ERROR\_ENTRY(110,"Ошибка при открытии файла с исходным кодом (-in)"),  ERROR\_ENTRY(111,"Недопустимый символ в исходном файле (-in)"),  ERROR\_ENTRY(112,"Ошибка при создании файла протокола (-log)"),  ERROR\_ENTRY\_NODEF(113), ERROR\_ENTRY\_NODEF(114),ERROR\_ENTRY\_NODEF(115),  ERROR\_ENTRY\_NODEF(116), ERROR\_ENTRY\_NODEF(117),ERROR\_ENTRY\_NODEF(118),ERROR\_ENTRY\_NODEF(119),  ERROR\_ENTRY\_NODEF10(120), ERROR\_ENTRY\_NODEF10(130), ERROR\_ENTRY\_NODEF10(140),ERROR\_ENTRY\_NODEF10(150),  ERROR\_ENTRY\_NODEF10(160), ERROR\_ENTRY\_NODEF10(170), ERROR\_ENTRY\_NODEF10(180),ERROR\_ENTRY\_NODEF10(190),  ERROR\_ENTRY\_NODEF100(200), ERROR\_ENTRY\_NODEF100(300), ERROR\_ENTRY\_NODEF100(400), ERROR\_ENTRY\_NODEF100(500),  ERROR\_ENTRY\_NODEF100(600), ERROR\_ENTRY\_NODEF100(700), ERROR\_ENTRY\_NODEF100(800), ERROR\_ENTRY\_NODEF100(900)  };  ERROR geterror(int id)  {  if (id<0 || id>ERROR\_MAX\_ENTRY)  {  return errors[0];  }  else  {  return errors[id];  }  }  ERROR geterrorin(int id, int line = 0, int col = 0)  {  if (id<0 || id>ERROR\_MAX\_ENTRY)  {  ERROR f;  f = errors[0];  f.inext.line = line;  f.inext.col = col;  return f;  }  else  {  ERROR f;  f = errors[id];  f.inext.line = line;  f.inext.col = col;  return f;  }  }  }; |
| Содержимое файла Error.cpp |

|  |
| --- |
| #pragma once  #define ERROR\_THROW(id) Error::geterror(id); // throw ERROR\_THROW(id)  #define ERROR\_THROW\_IN(id,l,c) Error::geterrorin(id,l,c); // throw ERROR\_THROW(id, строка, колонка)  #define ERROR\_ENTRY(id,m) {id,m,{-1,-1}} // элемент таблицы ошибок  #define ERROR\_MAXSIZE\_MESSAGE 200 // максимальная длина сообщения о ошибке  #define ERROR\_ENTRY\_NODEF(id) ERROR\_ENTRY(-id,"Неопределённая ошибка") // 1 неопредлённый элемент таблицы ошибок  #define ERROR\_ENTRY\_NODEF10(id) ERROR\_ENTRY\_NODEF(id+0),ERROR\_ENTRY\_NODEF(id+1),ERROR\_ENTRY\_NODEF(id+2),\  ERROR\_ENTRY\_NODEF(id+3),ERROR\_ENTRY\_NODEF(id+4),ERROR\_ENTRY\_NODEF(id+5),\  ERROR\_ENTRY\_NODEF(id+6),ERROR\_ENTRY\_NODEF(id+7),ERROR\_ENTRY\_NODEF(id+8),ERROR\_ENTRY\_NODEF(id+9)  #define ERROR\_ENTRY\_NODEF100(id) ERROR\_ENTRY\_NODEF10(id+0),ERROR\_ENTRY\_NODEF10(id+10),ERROR\_ENTRY\_NODEF10(id+20),ERROR\_ENTRY\_NODEF10(id+30), \  ERROR\_ENTRY\_NODEF10(id+40),ERROR\_ENTRY\_NODEF10(id+50),ERROR\_ENTRY\_NODEF10(id+60),ERROR\_ENTRY\_NODEF10(id+70), \  ERROR\_ENTRY\_NODEF10(id+80),ERROR\_ENTRY\_NODEF10(id+90)  #define ERROR\_MAX\_ENTRY 1000 // количество элементов в таблице ошибок  namespace Error  {  struct ERROR  {  int id; // код ошибки  char message[ERROR\_MAXSIZE\_MESSAGE]; // сообщение о ошибке  struct IN // расширение для ошибок при входных данных  {  short line; // номер строки  short col; // номер позиции в строке  } inext;  };  ERROR geterror(int id);  ERROR geterrorin(int id, int line, int col);  }; |
| Содержимое файла Error.h |

# Приложение Б

|  |
| --- |
| ##pragma once  #include "stdafx.h"  #define PARM\_IN L"-in:" // ключ для файла исходного кода  #define PARM\_OUT L"-out:" // ключ для файла обьектного кода  #define PARM\_LOG L"-log:" // ключ для файла журнала  #define PARM\_MAX\_SIZE 300 // макс длина строки параметра  #define PARM\_OUT\_DEFAULT\_EXT L".out" // расширение файла обьектного кода по умолчанию  #define PARM\_LOG\_DEFAULT\_EXT L".log" // расширение файла протокола по умолчанию  namespace Parm // обработка входных параметров  {  struct PARM // входные параметры  {  wchar\_t in[PARM\_MAX\_SIZE]; // имя файла исходного кода  wchar\_t out[PARM\_MAX\_SIZE]; // имя файла обьектного кода  wchar\_t log[PARM\_MAX\_SIZE]; // имя файла протокола  };  PARM getparm(int argc, \_TCHAR\* argv[]);  }; |
| Содержимое файла Parm.h |

|  |
| --- |
| #include "stdafx.h"  #include "Parm.h"  #include "Error.h"  namespace Parm  {  PARM getparm(int argc, \_TCHAR\* argv[])  {  bool bin = 0, bout = 0, blog = 0;  PARM f;  for (int l = 0; l < argc; l++)  {  if (wcsstr(argv[l], PARM\_IN))  {  bin = 1;  }  if (wcsstr(argv[l], PARM\_OUT))  {  bout = 1;  }  if (wcsstr(argv[l], PARM\_LOG))  {  blog = 1;  }  }  if (bin == 0)  {  throw ERROR\_THROW(100)  }  for (int i = 0; i < argc; i++)  {  if (wcslen(argv[i]) > PARM\_MAX\_SIZE) throw ERROR\_THROW(104);  if (wcsstr(argv[i], PARM\_IN))  {  wchar\_t\* h = wcsstr(argv[i], L":") + 1;  wcscpy\_s(f.in, h);  }  if (wcsstr(argv[i], PARM\_OUT))  {  wchar\_t\* h = wcsstr(argv[i], L":") + 1;  wcscpy\_s(f.out, h);  }  if (wcsstr(argv[i], PARM\_LOG))  {  wchar\_t\* h = wcsstr(argv[i], L":") + 1;  wcscpy\_s(f.log, h);  }  }  if (bout == 0)  {  wcscpy\_s(f.out, f.in);  wcscat\_s(f.out, PARM\_OUT\_DEFAULT\_EXT);  }  if (blog == 0)  {  wcscpy\_s(f.log, f.in);  wcscat\_s(f.log, PARM\_LOG\_DEFAULT\_EXT);  }  return f;  }  }; |
| Содержимое файла Parm.cpp |

# Приложение С

|  |
| --- |
| #pragma once  #define IN\_MAX\_LEN\_TEXT 1024\*1024 // макс размер исходного кода = 1МБ  #define IN\_CODE\_ENDL '\n' // символ конца строки  // таблица проверки входной информации, индекс = код (1251) символов  // значения IN::F - запрещённый символ, IN::T - разрешённый символ, IN::I - игнорировать (не вводить)  // если 0 <= значение < 256 - то вводится данное значение  #define IN\_CODE\_TABLE {\  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::T, IN::F, IN::F, IN::I, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::T, IN::T, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::T, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::T, IN::I, '!', IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::T, IN::T, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::T, IN::F, IN::F, IN::T, \  IN::F, IN::F, IN::T, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::T, IN::F, IN::F, IN::T, \  IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::F, IN::F, IN::T, IN::F, IN::F, IN::T, IN::F, IN::F, IN::T, IN::T, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, \  IN::T, IN::F, IN::T, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F, IN::F \  }  namespace In  {  struct IN // исходные данные  {  enum { T = 1024, F = 2048, I = 4096 }; // Т - допустимый символ, F - недопустимый, I - игнорировать, иначе заменить  int size; // размер исходного кода  int lines; // количество строк  int ignor; // количество проигнорированных символов  unsigned char\* text; // исходный код (1251)  int code[256] = IN\_CODE\_TABLE; // таблица проверки  };  IN getin(wchar\_t infile[]);  }; |
| Содержимое файла In.h |

|  |
| --- |
| #include "stdafx.h"  #include "In.h"  #include "Error.h"  #include <iostream>  #include <fstream>  namespace In  {  IN getin(wchar\_t infile[])  {  IN f;  unsigned char ch;  static unsigned char l[1024] = "";  int line = 1, position = 1, size = 0, ignor = 0, i = 0;  std::ifstream in(infile);  if (in.fail() == 0)  {  ch = in.get();  while (!in.eof())  {  if (ch == ((unsigned char)'\n'))  {  line++;  position = 0;  }  if (f.code[(int)ch] == IN::T)  {  position++;  l[i] = ch;  i++;  }  else if (f.code[(int)ch] == IN::F)  {  throw ERROR\_THROW\_IN(111, line, position);  }  else if (f.code[(int)ch] == IN::I)  {  ignor++;  }  else  {  l[i] = f.code[ch];  i++;  }  ch = in.get();  }  in.close();  f.size = i;  f.text = l;  f.lines = line;  f.ignor = ignor;  }  else  {  throw ERROR\_THROW(110);  }  return f;  }  }; |
| Содержимое файла In.cpp |

# Приложение Д

|  |
| --- |
| #pragma once  #include <fstream>  #include "In.h"  #include "Parm.h"  #include "Error.h"  namespace Log // Работа с протоколом  {  struct LOG // протокол  {  wchar\_t logfile[PARM\_MAX\_SIZE]; // имя файла протокола  std::ofstream\* stream; // выходной поток протокола  };  static const LOG INITLOG = { L"",NULL }; // структура для начальной инициализации LOG  LOG getlog(wchar\_t logfile[]); // сформировать структуру LOG  void WriteLine(LOG log, char\* c, ...); // вывести в протокол конкатенацию строк  void WriteLine(LOG log, wchar\_t\* c, ...);// вывести в протокол конкатенацию строк  void WriteLog(LOG log); // вывести в протокол заголовок  void WriteParm(LOG log, Parm::PARM parm);// вывести в протокол информацию о входных параметрах  void WriteIn(LOG log, In::IN in); // вывести в протокол информацию о входном потоке  void WriteError(LOG log, Error::ERROR error); // вывести в протокол информацию об ошибке  void Close(LOG log);  }; |
| Содержимое файла Log.h |

|  |
| --- |
| #include "stdafx.h"  #include "Log.h"  #include "Parm.h"  #include "In.h"  #include <iostream>  #include <ctime>  namespace Log  {  LOG getlog(wchar\_t logfile[])  {  static std::ofstream stream(logfile);  stream.seekp(0);  LOG f;  f.stream = &stream;  wcscpy\_s(f.logfile, logfile);  f.stream->is\_open();  if (f.stream->fail())  {  throw ERROR\_THROW(112);  }  return f;  }  void WriteLine(LOG log, char\* c, ...)  {  log.stream->is\_open();  char\*\* f;  f = &c;  char l[256] = "";  while (\*f != "")  {  strcat\_s(l, \*f);  f = f + 1;  }  \*(log.stream) << l;  }  void WriteLine(LOG log, wchar\_t\* c, ...)  {  wchar\_t\*\* p = &c;  char answ[100] = "", buf[100] = "";  while (\*p != L"")  {  wcstombs\_s(0, buf, \*p, 100);  strcat\_s(answ, buf);  \*(p)++;  }  \*(log.stream) << answ;  }  void WriteLog(LOG log)  {  time\_t rawtime;  char buffer[80];  time(&rawtime);  struct tm timeinfo;  localtime\_s(&timeinfo, &rawtime);  strftime(buffer, 80, "%d.%m.%Y %H:%M:%S", &timeinfo);  if (log.stream->fail() == 0)  {  log.stream->write("---Протокол--- ", sizeof("---Протокол--- "));  \*(log.stream) << buffer << "----------" << std::endl;  }  }  void WriteParm(LOG log, Parm::PARM parm)  {  if (log.stream->fail() == 0)  {  char m[512] = "";  log.stream->write("---Параметры---", sizeof("---Параметры---"));  (\*log.stream) << std::endl;  wcstombs\_s(0, m, parm.in, sizeof(parm.in));  \*(log.stream) << "-in: " << m << std::endl;  wcstombs\_s(0, m, parm.out, sizeof(parm.out));  \*(log.stream) << "-out: " << m << std::endl;  wcstombs\_s(0, m, parm.log, sizeof(parm.log));  \*(log.stream) << "-log: " << m << std::endl;  }  else  {  std::cout << "in: " << parm.in << std::endl;  std::cout << "out:" << parm.out << std::endl;  std::cout << "log: " << parm.log << std::endl;  }  }  void WriteIn(LOG log, In::IN in)  {  if (log.stream->fail() == 0)  {  log.stream->write("---Исходные данные---", sizeof("---Исходные данные---"));  (\*log.stream) << std::endl;  log.stream->write("Количество символов:", sizeof("Количество символов:"));  \*(log.stream) << in.size << std::endl;  log.stream->write("Проигнорировано: ", sizeof("Проигнорировано: "));  \*(log.stream) << in.ignor << std::endl;  log.stream->write("Количество строк: ", sizeof("Количество строк: "));  \*(log.stream) << in.lines << std::endl;  }  else  {  std::cout << "Количество символов: " << in.size << std::endl;  std::cout << "Проигнорировано:" << in.ignor << std::endl;  std::cout << "Количество строк: " << in.lines << std::endl;  }  }  void WriteError(LOG log, Error::ERROR error)  {  if (log.stream->fail() == 0)  {  \*(log.stream) << error.id << " " << error.message << " строка " << error.inext.line << " позиция " << error.inext.col;  \*(log.stream) << std::endl;  }  else  {  std::cout << error.id << " " << error.message << " " << "строка" << error.inext.line << " позиция" << error.inext.col << std::endl;  }  }  void Close(LOG log)  {  log.stream->close();  }  }; |
| Содержимое файла Log.cpp |

# Приложение Е

|  |
| --- |
| #include "stdafx.h"  #include <iostream>  #include <locale>  #include <cwchar>  #include "Error.h"  #include "Parm.h"  #include "Log.h"  #include "In.h"  int \_tmain(int argc, \_TCHAR\* argv[])  {  setlocale(LC\_CTYPE, "rus");  // ТЕСТ ERROR  std::cout << "---- тест Error::geterror ----" << std::endl << std::endl;  try  {  throw ERROR\_THROW(100);  }  catch (Error::ERROR e)  {  std::cout << "Ошибка " << e.id << ": " << e.message << std::endl << std::endl;  };  try  {  throw ERROR\_THROW\_IN(111, 7, 7);  }  catch (Error::ERROR e)  {  std::cout << "Ошибка " << e.id << ": " << e.message  << ", строка " << e.inext.line  << ", позиция " << e.inext.col << std::endl << std::endl;  };  // ТЕСТ PARM  std::cout << "---- тест Parm::getparm ----" << std::endl << std::endl;  try  {  Parm::PARM parm = Parm::getparm(argc, argv);  std::wcout << "-in:" << parm.in << ", -out:" << parm.out << ", -log:" << parm.log << std::endl << std::endl;  }  catch (Error::ERROR e)  {  std::cout << "Ошибка " << e.id << ": " << e.message << std::endl << std::endl;  };  // ТЕСТ IN 1  std::cout << "---- тест In::getin 1 ----" << std::endl << std::endl;  try  {  Parm::PARM parm = Parm::getparm(argc, argv);  In::IN in = In::getin(parm.in);  std::cout << in.text << std::endl;  std::cout << "Всего символов: " << in.size << std::endl;  std::cout << "Всего строк: " << in.lines << std::endl;  std::cout << "Пропущено: " << in.ignor << std::endl << std::endl;  }  catch (Error::ERROR e)  {  std::cout << "Ошибка " << e.id << ": " << e.message << std::endl << std::endl;  };  // ТЕСТ IN 2  std::cout << "---- тест In::getin 2 ----" << std::endl << std::endl;  try  {  Parm::PARM parm = Parm::getparm(argc, argv);  In::IN in = In::getin(parm.in);  std::cout << in.text << std::endl;  std::cout << "Всего символов: " << in.size << std::endl;  std::cout << "Всего строк: " << in.lines << std::endl;  std::cout << "Пропущено: " << in.ignor << std::endl << std::endl;  }  catch (Error::ERROR e)  {  std::cout << "Ошибка " << e.id << ": " << e.message << std::endl << std::endl;  std::cout << "строка " << e.inext.line << " позиция " << e.inext.col << std::endl << std::endl;  };  // ТЕСТ LOG  std::cout << "---- тест LOG::log ----" << std::endl << std::endl;  Log::LOG log = Log::INITLOG;  try  {  Parm::PARM parm = Parm::getparm(argc, argv);  log = Log::getlog(parm.log);  Log::WriteLine(log, (char\*)"Тест:", (char\*)" без ошибок \n", "");  Log::WriteLine(log, (wchar\_t\*)L"Тест:", (wchar\_t\*)L" без ошибок \n", L"");  Log::WriteLog(log);  Log::WriteParm(log, parm);  In::IN in = In::getin(parm.in);  Log::WriteIn(log, in);  Log::Close(log);  }  catch (Error::ERROR e)  {  Log::WriteError(log, e);  };  system("pause");  return 0;  } |
| Содержимое файла SE\_Lab15.cpp |