# ЛАБОРАТОРНАЯ РАБОТА №7

# ИССЛЕДОВАНИЕ СРЕДСТВ УПРАВЛЕНИЯ ПОТОКАМИ ВВОДА-ВЫВОДА. ИССЛЕДОВАНИЕ МЕХАНИЗМА ОБРАБОТКИ ИСКЛЮЧЕНИЙ

Цель работы

Изучить способы реализации и особенности управления потоками ввода/вывода, исследовать способы генерации и обработки исключений.

Задания

1. В ходе самостоятельной подготовки изучить основы работы с потоковым вводом-выводом и исключениями;
2. Разработать согласно варианту (Рисунок 1) программу на языке C++, состоящую из двух частей: первая демонстрирует умение управлять потоками ввода-вывода, вторая демонстрирует умение генерировать и перехватывать исключения;

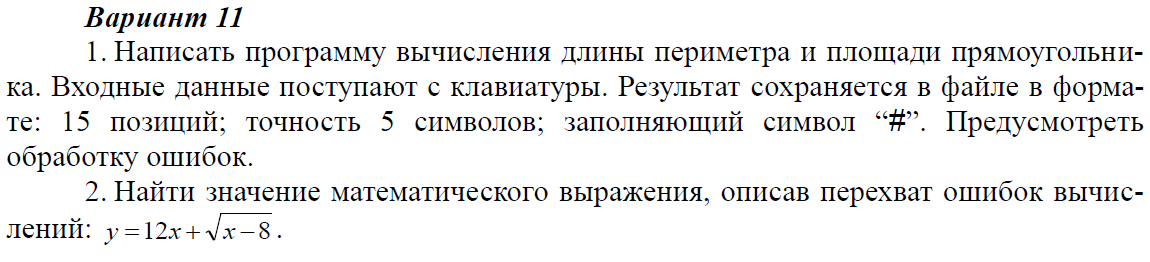


Рисунок 1 – Вариант задания

1. Разработать тестовые примеры и выполнить тестирование программы;
2. Получить результаты работы программы и исследовать её свойства для различных режимов работы, сформулировать выводы;

Текст программы

#include <iostream>

#include <fstream>

#include <cmath>

#include <string>

#include <limits>

using namespace std;

// Класс исключений

class MyException

{

public:

string message;

MyException(string m)

{

message = m;

}

};

// Ввод double-числа с проверкой

int enterDub(double x)

{

double n;

cin >> n;

if (!cin)

{

cin.clear();

cin.ignore(INT\_MAX, '\n');

throw new MyException("Error: wrong value");

}

cin.ignore(INT\_MAX, '\n');

return n;

}

// Форматированный ввод значений в файл

void toFile(double p, double s)

{

ofstream fout("lab7.txt");

if (!fout)

{

throw new MyException("Error: Can't open file");

}

fout.precision(5);

fout.fill('#');

fout.width(15);

fout << p << endl;

fout.width(15);

fout << s << endl;

fout.close();

return;

}

// Вычисление формулы с проверкой потоков

void formula(double x)

{

if (x < 8) throw new MyException("Error: x should be >= 8");

cout << "y = " << 12 \* x + sqrt(x - 8) << endl;

return;

}

int main()

{

// Проверка ввода в файл

double l = 0, w = 0, p, s;

try

{

cout << "Enter length - ";

l = enterDub(l);

cout << "Enter width - ";

w = enterDub(w);

s = w \* l;

p = 2 \* (w + l);

if ((l <= 0) || (w <= 0)) throw new MyException("Error: value <= 0");

toFile(p, s);

}

catch (MyException\* a)

{

cout << a->message << endl;

delete a;

}

catch (...)

{

cout << "Unexpected error" << endl;

}

// Проверка перехвата ошибок

cout << endl;

double x, y;

cout << "Enter x - ";

cin >> x;

try {

formula(x);

}

catch (MyException\* a)

{

cout << a->message << endl;

delete a;

}

catch (...)

{

cout << "Unexpected error" << endl;

}

}

Тестовые примеры

В ходе тестирования программа выполнила обработку значений, удовлетворяющих её требованиям, а при неверных значениях правильно определила и вывела суть возникших ошибок (Рисунок 2, Рисунок 3).

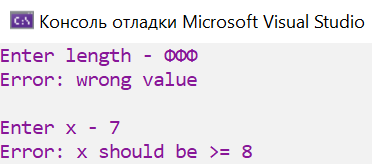


Рисунок 2 – Проверка на обработку ошибок

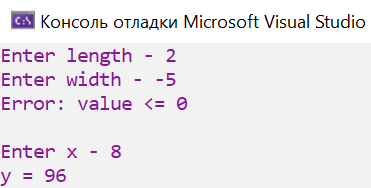


Рисунок 3 – Тестирование программы

При вводе после вещественных чисел посторонних символов (запятой и восклицательного знака) программа занесла в переменную считываемую часть, остальные символы были проигнорированы потоком ввода (Рисунок 4).

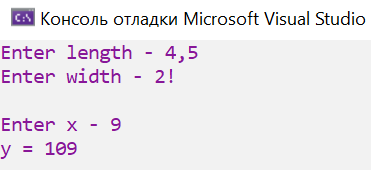


Рисунок 4 – Тестирование программы

В результате в файл были введены отформатированные согласно заданию значения периметра и площади прямоугольника (Рисунок 5).

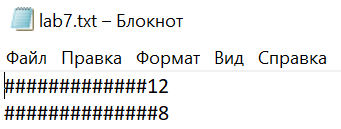


Рисунок 5 – Результат записи в файл

Вывод

В ходе работы были изучены особенности работы и способы реализации потоков ввода-вывода в объектно-ориентированном программировании. Освоены методы работы с потоковыми классами, механизмы форматирования вывода с помощью флагов и манипуляторов, обработка исключительных состояний в потоках. Также был изучен механизм обработки исключений и его возможно.

На основе полученных знаний была написана программа, в которой реализована проверка вводимых значений и форматированная запись информации в файл через файловый поток.