

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.03 | Прикладная информатика | | |
| Направленность (профиль) | |  | Прикладная информатика в химии | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Программирование | | |
| Курс | I | | | Группа | 415 |

Отчёт по контрольной работе № 2

Вариант №5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 415 |  |  |  | Никита Леванович Шарашидзе |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Федин Алексей Константинович |
|  |  | (дата, подпись) |  | Корниенко Иван Григорьевич |

Содержание

[1 Постановка задачи 3](#_Toc99067513)

[2 Исходные данные 3](#_Toc99067514)

[3 Особые ситуации 3](#_Toc99067515)

[4 Математические методы и алгоритмы решения задач 3](#_Toc99067516)

[5 Форматы представления данных 4](#_Toc99067517)

[6 Структура программы 4](#_Toc99067518)

[7 Блок-схемы алгоритмов 7](#_Toc99067519)

[8 Результаты работы программы 9](#_Toc99067522)

[9 Описание хода выполнения контрольной работы 14](#_Toc99067523)

[10 Исходный текст программы 14](#_Toc99067524)

## 1 Постановка задачи

Создать базовый класс фигура, производные классы: круг, прямоугольник, трапеция. Определить виртуальные функции для вычисления площади и периметра.

## 2 Исходные данные

В качестве исходных данных программа использует список характеристик фигур, введённый пользователем. Для прямоугольника длинна и ширина, для круга радиус, для трапеции это длинна оснований и сторон, а также высота.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

– если пользователь ввёл характеристики фигур не корректно результаты не могут быть получены,

– если программа не может получить доступ к исходным файлам и информации в ним в нужной форме, то работа программы невозможна.

## 4 Математические методы и алгоритмы решения задач

Согласно постановке задачи для составления программы будет использованы два алгоритма реализующие:

-получение значения площади фигуры

-получение значения периметра фигуры

## 5 Форматы представления данных

Таблица 1- Переменные, используемые в программе

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| counter | int | Счётчик количества фигур |
| user\_choice | int | Выбор пользователя |
| is\_stream\_opened | bool | Проверка на ввод данных из файла |
| name | string | Переменная класса, хранящая названия фигуры |
| width\_ | double | Переменная класса, хранящая данные о длине стороны прямоугольника |
| height\_ | double | Переменная класса, хранящая данные о ширине стороны прямоугольника |
| large\_base\_ | double | Переменная класса, хранящая данные о большем основании трапеции |

Продолжение таблицы 1

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Описание |
| smaller\_base\_ | double | Переменная класса, хранящая данные о меньшем основании  Трапеции |
| right\_side\_ | double | Переменная класса, хранящая данные о правой стороне  трапеции |
| left\_side\_ | double | Переменная класса, хранящая данные о левой стороне  трапеции |
| radius\_ | double | Переменная класса, хранящая данные о радиусе круга |

Для выбора вариаций алгоритма и пунктов меню используются константные выражения

Таблица 2- Константные значения, используемые в программе

|  |  |  |  |
| --- | --- | --- | --- |
| Имя | Тип | Значение | Описание |
| \_rectangle | enum figureChoise | 1 | Пункт меню, ответственный за выбор прямоугольника |
| \_circle | enum figureChoise | 2 | Пункт меню, ответственный за выбор круга |
| \_trapezoid | enum figureChoise | 3 | Пункт меню, ответственный за выбор трапеции |
| file | enum startChoise | 1 | Пункт меню, ответственный за выбор файлового ввода |
| console | enum startChoise | 2 | Пункт меню, ответственный за выбор консольного ввода |

Данные могут быть предоставлены двумя видами, с помощью консольного ввода и в виде файла с разрешением “.txt”, где каждая характеристика записана в отдельную строку.

## 6 Структура программы

Для оптимизации работы с кодом было принято решение разделить программу на модули.

|  |  |
| --- | --- |
| Имя | Описание |
| Program.cpp | Основной модуль |
| Class.cpp | Модуль включающий в себя базовый и наследующие классы |
| Interface.cpp | Модуль, включающий в себя модульное меню |
| Algoritm.cpp | Модуль, включающий в себя алгоритмы выполнения задания |
| Iofile.cpp | Модуль, ответственный за работу с файлами |
| Check.cpp | Модуль, ответственный за обработку исключений |

Таблица 3-Таблица модулей, составляющих программу

Каждый модуль разделён на функции, выполняющие какие-либо задачи.

Таблица 4-Таблица функций, составляющих модуль “Algoritm.cpp”.

|  |  |
| --- | --- |
| Имя | Описание |
| rectangleChoice | Функция, производящая расчёт параметров прямоугольника |
| circleChoice | Функция, производящая расчёт параметров круга |
| trapezoidChoice | Функция, производящая расчёт параметров трапеции |
| showResult | Функция, выводящая результаты работы на экран |

Таблица 5-Таблица функций, составляющих модуль “ Interface.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| showFigureMenu | Функция, выводящая список существующих алгоритмов |
| ShowGreetings | Функция, выводящая приветствие |
| ShowTask | Функция, выводящая исходное задание |
| ShowInputChoice | Меню выбора типа ввода данных |
| showFinOutputMenu | Меню вывода результата |
| showOutputMenu | Меню сохранения введённых данных |
| showEndMenu | Меню выбора окончания программы |
| showOutputMenu | Меню взаимодействия с файлом |
| showContMenu | Меню добавления фигур |

Таблица 6- Таблица функций, составляющих модуль “ Figure.h”.

|  |  |
| --- | --- |
| Имя функции | Описание |
| GetSquare | Виртуальная функция нахождения площади |
| GetPerimeter | Виртуальная функция нахождения периметра |
| GetName | Функция, возвращающая переменную “имени” |
| Имя класса | Описание |
| figure | Базовый класс |
| circle | Класс “circle” наследуемый от “figure” |
| rectangle | Класс “ rectangle” наследуемый от “figure” |
| trapezoid | Класс “ trapezoid” наследуемый от “figure” |

Таблица 7- Таблица функций и переменных, составляющих модуль “Check.h”.

|  |  |
| --- | --- |
| Имя функции | Описание |
| getPositiveValue | Функция проверки корректности ввода положительного числа |
| CheckLineS | Функция проверки корректности ввода строки |
| CheckLineD | Функция проверки корректности ввода строки со значением числа с плавающей запятой |
| CheckLineI | Функция проверки корректности ввода строки c целочисленным значением |
| getStartChoice | Функция проверки корректности пользовательского ввода при старте программы |
| getBinChoice | Функция проверки корректности пользовательского ввода ДА/НЕТ |
| getFigureChoice | Функция проверки корректности пользовательского ввода при выборе пункта меню |
| checkFileName | Функция проверки файла на возможность создания |
| checkFileExistance | Функция проверки существования файла |

Таблица 8- Таблица функций и переменных, составляющих модуль “ File.h”.

|  |  |
| --- | --- |
| Имя | Описание |
| FileOutput | Функция вывода исходных данных в файл |
| FileInput | Функция ввода данных из файла |
| FileOutputFin | Функция вывода данных в файл |

## 

## 7 Блок-схемы алгоритмов

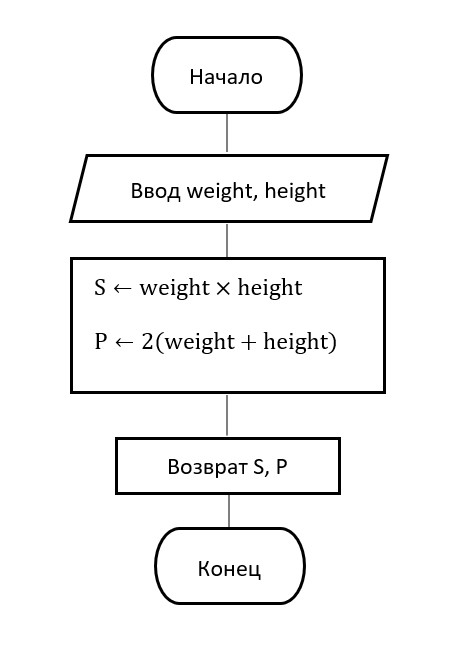


Рисунок 1-Блок-схема алгоритма функции нахождения периметра и площади прямоугольника

## 

Рисунок 2- Блок-схема алгоритма функции нахождения периметра и площади круга

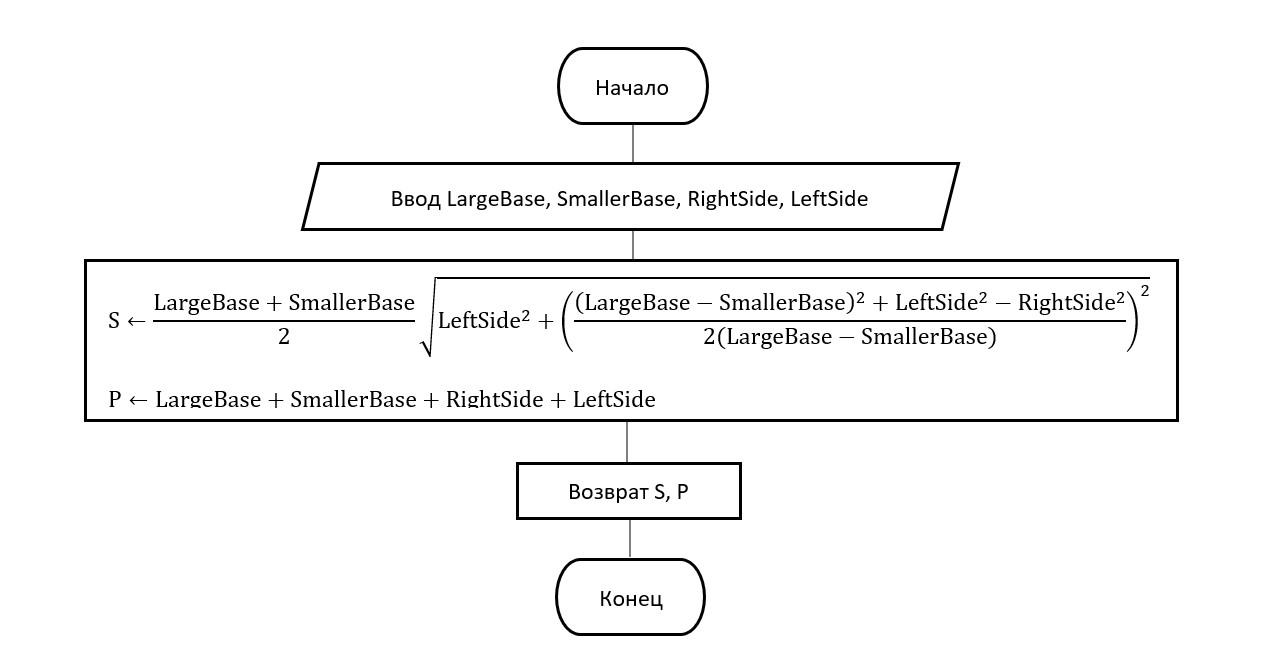


Рисунок 3- Блок-схема алгоритма функции нахождения периметра и площади трапеции

## 8 Результаты работы программы

В результате работы программы создаётся вектор указателей на объекты класса figure, данные которого могут быть выведены на консоль либо сохранены в файл. На экранных копиях работы программы показаны как корректный ввод данных, так и не корректный.

Экранные копии работы программы:

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4-Экранная копия результатов работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5-Экранная копия результатов работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6-Экранная копия результатов работы программы

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 7-Экранная копия результатов работы программы

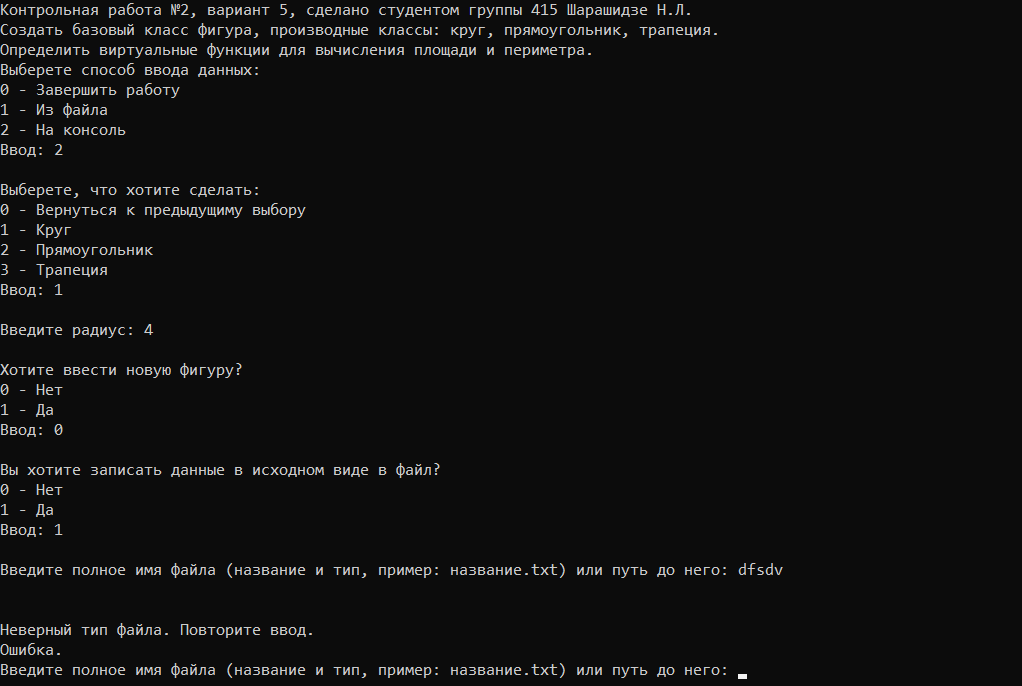


Рисунок 8-Экранная копия результатов работы программы

## 9 Описание хода выполнения контрольной работы

1) В ходе контрольной работы было создано решение (Solution) в

интегрированной среде разработки Microsoft Visual Studio C++ 2017. В нём был

создан проект.

2) После набора текста программы выяснилось, что вывод текста на экран

консольного приложения работает неправильно из-за различия кодировок

консольного приложения и среды разработки. Для решения этой проблемы

была использована функция setlocale(LC\_ALL, "Russian"), которая

обеспечивает работу приложения с символами кириллицы. Но также была выявлена проблема с записью символов кириллицы в переменные типа string. Для решения этой проблемы была подключена <windows.h>, и использованы две функции SetConsoleCP(1251), SetConsoleOutputCP(1251);

3) При написании блока ввода данных было принято решение использовать функцию getline, но при тестировании была выявлена проблема загрязнения потока ввода остаточными данными. У данной проблемы были найдены решения, первое-ручное очищение потока ввода с помощью функций cin.clear() и cin.ignore(), но данное решение не было практичным и усложняло читаемость кода. В итоге было принято решение использовать функцию cin из имён пространства std, но возможность реализации ввода через getline может быть осуществлена.

## 10 Исходный текст программы

1)Main.cpp

// main.cpp

// Контрольная работа No 2.

// Использование языка С++ для математических расчётов

//Создать базовый класс фигура, производные классы: круг, прямоугольник,

//трапеция. Определить виртуальные функции для вычисления площади и периметра.

// Студент группы 415, Никита Леванович Шарашидзе. 2022 год

#include "program.h"

int main(void)

{

program();

}

2)Algoritm.h

#pragma once

#include <iostream>

#include <vector>

#include <memory>

#include "class.h"

#include "interface.h"

#include "iofile.h"

#include "check.h"

void circleChoice(std::vector<std::shared\_ptr<figure>>& shp);

void rectangleChoice(std::vector<std::shared\_ptr<figure>>& shp);

void trapezoidChoice(std::vector<std::shared\_ptr<figure>>& shp);

void showResult(std::vector<std::shared\_ptr<figure>>& shp);

2.1) Algoritm.cpp

#include "algorithm.h"

void circleChoice(std::vector<std::shared\_ptr<figure>>& shp)

{

auto temp = std::make\_shared<circle>();

std::cout << "Введите радиус: ";

temp->setRadius(getPositiveValue<double>());

shp.push\_back(temp);

}

void rectangleChoice(std::vector<std::shared\_ptr<figure>>& shp)

{

auto temp = std::make\_shared<rectangle>();

std::cout << "Введите высоту: ";

temp->setHeight(getPositiveValue<double>());

std::cout << "Введите ширину: ";

temp->setWidth(getPositiveValue<double>());

shp.push\_back(temp);

}

void trapezoidChoice(std::vector<std::shared\_ptr<figure>>& shp)

{

while (true) {

auto temp = std::make\_shared<trapezoid>();

std::cout << "Введите значение большего основания: ";

temp->setLargeBase(getPositiveValue<double>());

std::cout << "Введите значение левой стороны: ";

temp->setLeftSide(getPositiveValue<double>());

std::cout << "Введите значение правой стороны: ";

temp->setRightSide(getPositiveValue<double>());

std::cout << "Введите значение меньшего основания: ";

temp->setSmallerBase(getPositiveValue<double>());

if (!(temp->checkTrapezoid(temp->getLargeBase(), temp->getLeftSide(), temp->getRightSide(), temp->getSmallerBase())))

{

std::cout << "Введены некорректные данные: ";

temp->getSmallerBase() > temp->getLargeBase()

? std::cout << "меньшее основание больше большего основания." << std::endl

: std::cout << "длина одной из сторон больше суммы длин остальных сторон." << std::endl;

continue;

}

shp.push\_back(temp);

break;

}

}

void showResult(std::vector<std::shared\_ptr<figure>>& shp)

{

int index = 1;

std::cout << "Результат работы программы:" << std::endl;

for (auto& elem : shp) {

std::cout << "Объект № " << index << " Название фигуры: ";

std::cout << elem->getName() << std::endl;

std::cout << "Площадь: " << elem->getSquare() << std::endl;

std::cout << "Периметр: " << elem->getPerimeter() << std::endl;

std::cout << "" << std::endl;

index++;

}

}

3)Check.h

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include <cctype>

#include <algorithm>

#undef max

#define WIN\_NAME\_SIZE 22

#define QUIT 0

enum binChoise { no = 0, yes = 1 };

enum startChoise { file = 1, console = 2 };

enum figureChoise { \_circle = 1, \_rectangle = 2, \_trapezoid = 3 };

const std::string op = "output";

const std::string ip = "input";

const std::string txt = ".txt";

const std::string win\_name[WIN\_NAME\_SIZE] = { "CON", "PRN", "AUX", "NUL",

"COM1", "COM2", "COM3", "COM4", "COM5",

"COM6", "COM7", "COM8", "COM9", "LPT1",

"LPT2", "LPT3", "LPT4", "LPT5", "LPT6",

"LPT7", "LPT8", "LPT9" };

const int quit = 0;

const int error = 1;

int getStartChoice();

int getBinChoice();

int getFigureChoice();

int getLineI(std::ifstream& file);

double getLineD(std::ifstream& file);

std::string getLineS(std::ifstream& file);

template<typename T>

T getPositiveValue()

{

T temp = 0.0;

do

{

try

{

std::cin >> temp;

std::cout << std::endl;

if (std::cin.peek() != '\n') throw std::invalid\_argument("");

if (temp <= 0) throw std::out\_of\_range("");

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

return temp;

}

catch (const std::out\_of\_range&)

{

std::cout << "Число должно быть положительным. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

catch (const std::exception&)

{

std::cout << "Введены некорректные данные. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

}

} while (true);

}

void checkFileName(std::string str);

void checkFileExistance(std::string file\_name, const std::string stream\_type);

3.1)Check.cpp

#include "check.h"

int getStartChoice()

{

while (true)

{

try

{

int value = 0;

std::cin >> value;

std::cout << std::endl;

if (std::cin.peek() != '\n') throw std::invalid\_argument("");

if (value == quit || value == file || value == console)

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

return value;

}

else throw std::out\_of\_range("");

}

catch (const std::out\_of\_range&)

{

std::cout << "Такого пункта меню нет. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

continue;

}

catch (const std::exception&)

{

std::cout << "Некоректный тип данных. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

continue;

}

}

}

int getBinChoice()

{

while(true)

{

try

{

int value = 0;

std::cin >> value;

std::cout << std::endl;

if (std::cin.peek() != '\n') throw std::invalid\_argument("");

if (value == no || value == yes)

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

return value;

}

else throw std::out\_of\_range("");

}

catch (const std::out\_of\_range&)

{

std::cout << "Такого пункта меню нет. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

continue;

}

catch (const std::exception&)

{

std::cout << "Некоректный тип данных. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

continue;

}

}

}

int getFigureChoice()

{

int value = 0;

do {

try

{

value = 0;

std::cin >> value;

std::cout << std::endl;

if (std::cin.peek() != '\n') throw std::invalid\_argument("");

if (value == quit || value == \_circle || value == \_rectangle || value == \_trapezoid)

{

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

return value;

}

else throw std::out\_of\_range("");

}

catch (const std::out\_of\_range&)

{

std::cout << "Такого пункта в меню нет. Повторите ввод: " << std::endl;

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

continue;

}

catch (const std::exception&)

{

std::cout << "Некоректный тип данных. Повторите ввод: ";

std::cin.clear();

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n');

continue;

}

} while (true);

}

int getLineI(std::ifstream& file)

{

std::string temp\_s = "";

int temp\_i = 0;

try

{

std::getline(file, temp\_s);

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

throw error;

}

try

{

temp\_i = stoi(temp\_s);

if (temp\_i <= 0) throw error;

else return temp\_i;

}

catch (int)

{

std::cout << "Некорректный формат данных." << std::endl;

throw error;

}

}

double getLineD(std::ifstream& file)

{

std::string temp\_s = "";

double temp\_d = 0.0;

try

{

std::getline(file, temp\_s);

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

throw error;

}

try

{

temp\_d = stod(temp\_s);

if (temp\_d < 0) throw error;

else return temp\_d;

}

catch (int)

{

std::cout << "Некорректный формат данных." << std::endl;

throw error;

}

}

std::string getLineS(std::ifstream& file)

{

std::string temp\_1 = "";

try

{

std::getline(file, temp\_1);

return temp\_1;

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла." << std::endl;

throw error;

}

}

void checkFileName(std::string str)

{

std::cout << std::endl;

try

{

if (!(str.find(txt) == std::string::npos))

{

std::size\_t temp\_sz = str.find(txt);

std::string temp\_st = str.substr(temp\_sz);

if (temp\_st.size() != txt.size())

{

std::cout << "Неверный тип файла. Повторите ввод." << std::endl;

throw error;

}

}

else

{

std::cout << "Неверный тип файла. Повторите ввод." << std::endl;

throw error;

}

for (int i = 0; i < WIN\_NAME\_SIZE; i++)

{

std::string temp1 = win\_name[i] + txt;

std::string temp2 = str;

if (temp1 == temp2)

{

std::cout << "Введено зарезервированное операционной системой Windows слово. Повторите ввод." << std::endl;

throw error;

}

else if (temp1.size() == temp2.size())

{

std::transform(temp1.begin(), temp1.end(), temp1.begin(), ::toupper);

std::transform(temp2.begin(), temp2.end(), temp2.begin(), ::toupper);

if (temp1 == temp2)

{

std::cout << "Введено зарезервированное операционной системой Windows слово. Повторите ввод." << std::endl;

throw error;

}

}

}

}

catch (std::exception&)

{

std::cout << "Ошибка." << std::endl;

throw 1;

}

catch (int)

{

std::cout << "Ошибка." << std::endl;

throw 1;

}

}

void checkFileExistance(const std::string file\_name, const std::string stream\_type)

{

if (std::ifstream(file\_name))

{

if (stream\_type == op)

{

std::cout << "Файл уже существует. Вы хотите пересоздать его?" << std::endl

<< "0 - Нет." << std::endl

<< "1 - Да." << std::endl;

std::cout << "Ввод: ";

int userChoise = getBinChoice();

if (userChoise == !yes) throw 1;

}

}

else if (stream\_type == ip)

{

std::cout << "Файла с таким названием не существует. Потворите ввод." << std::endl;

throw 1;

}

}

4)class.h

#pragma once

#include <cmath>

#include <iostream>

#include <string>

const double m\_pi = acos(-1);

class figure {

public:

virtual double getSquare() = 0;

virtual double getPerimeter() = 0;

virtual std::string getName() = 0;

virtual ~figure() = default;

};

class circle : public figure {

private:

const std::string name\_ = "круг";

double radius\_ = 0;

public:

circle() = default;

circle(double radius);

void setRadius(double rad);

double getRadius();

double getSquare() override;

double getPerimeter() override;

std::string getName()override;

};

class rectangle : public figure {

private:

const std::string name\_ = "прямоугольник";

double width\_ = 0.0;

double height\_ = 0.0;

public:

rectangle() = default;

rectangle(double width, double height);

void setWidth(double width);

double getWidht();

void setHeight(double height);

double getHeight();

double getSquare() override;

double getPerimeter() override;

std::string getName()override;

};

class trapezoid : public figure {

private:

const std::string name\_ = "трапеция";

double large\_base\_ = 0.0;

double smaller\_base\_ = 0.0;

double right\_side\_ = 0.0;

double left\_side\_ = 0.0;

public:

trapezoid() = default;

trapezoid(double large\_base\_, double left\_side\_, double right\_side\_, double smaller\_base\_);

bool checkTrapezoid(double lb, double ls, double rs, double sb);

void setLargeBase(double lb);

double getLargeBase();

void setSmallerBase(double sb);

double getSmallerBase();

void setRightSide(double rs);

double getRightSide();

void setLeftSide(double ls);

double getLeftSide();

double getSquare() override;

double getPerimeter() override;

std::string getName()override;

};

4.1)class.cpp

#include "class.h"

circle::circle(double radius) :radius\_(radius) {}

void circle::setRadius(double rad)

{

radius\_ = rad;

}

double circle::getRadius()

{

return radius\_;

}

double circle::getSquare()

{

return m\_pi \* pow(radius\_, 2);

}

double circle::getPerimeter()

{

return 2 \* m\_pi \* radius\_;

}

std::string circle::getName()

{

return name\_;

}

rectangle::rectangle(double width, double height) :width\_(width), height\_(height)

{

}

void rectangle::setWidth(double width)

{

width\_ = width;

}

double rectangle::getWidht()

{

return width\_;

}

void rectangle::setHeight(double height)

{

height\_ = height;

}

double rectangle::getHeight()

{

return height\_;

}

double rectangle::getSquare()

{

return width\_ \* height\_;

}

double rectangle::getPerimeter()

{

return 2 \* (width\_ + height\_);

}

std::string rectangle::getName()

{

return name\_;

}

trapezoid::trapezoid(double largeBase, double smallerBase, double rightSide, double leftSide) :large\_base\_(largeBase), smaller\_base\_(smallerBase), right\_side\_(rightSide), left\_side\_(leftSide)

{

}

bool trapezoid::checkTrapezoid(double lb, double ls, double rs, double sb)

{

double side[4] = { lb, ls, rs, sb };

double sum = 0.0f;

for (int i = 0; i < 4; i++) sum += side[i];

for (int i = 0; i < 4; i++) if (side[i] >= sum - side[i]) return false;

if (fabs(lb - sb) <= 0) return false;

return true;

}

void trapezoid::setLargeBase(double lb)

{

large\_base\_ = lb;

}

double trapezoid::getLargeBase()

{

return large\_base\_;

}

void trapezoid::setSmallerBase(double sb)

{

smaller\_base\_ = sb;

}

double trapezoid::getSmallerBase()

{

return smaller\_base\_;

}

void trapezoid::setRightSide(double rs)

{

right\_side\_ = rs;

}

double trapezoid::getRightSide()

{

return right\_side\_;

}

void trapezoid::setLeftSide(double ls)

{

left\_side\_ = ls;

}

double trapezoid::getLeftSide()

{

return left\_side\_;

}

double trapezoid::getSquare()

{

return (((smaller\_base\_ + large\_base\_) / 2) \* (sqrt(pow(left\_side\_, 2)

- pow(((pow(smaller\_base\_ - large\_base\_, 2) + pow(left\_side\_, 2)

- pow(right\_side\_, 2)) / (2 \* smaller\_base\_ - 2 \* large\_base\_)), 2))));

}

double trapezoid::getPerimeter()

{

return left\_side\_ + right\_side\_ + large\_base\_ + smaller\_base\_;

}

std::string trapezoid::getName()

{

return name\_;

}

5) iofile.h

#pragma once

#include <memory>

#include <vector>

#include <fstream>

#include <filesystem>

#include "algorithm.h"

#include "check.h"

void fileOutput(std::vector<std::shared\_ptr<figure>>& shp, int counter);

void fileInput(std::vector<std::shared\_ptr<figure>>& shp);

void fileOutputFin(std::vector<std::shared\_ptr<figure>>& shp);

5) iofile.cpp

#include "iofile.h"

void fileOutput(std::vector<std::shared\_ptr<figure>>& shp, int counter)

{

std::ofstream file;

file.exceptions(std::ofstream::badbit | std::ofstream::failbit);

bool is\_stream\_opened = false;

std::string file\_name = "";

do

{

try

{

std::cout << "Введите полное имя файла (название и тип, пример: название.txt) или путь до него: ";

std::getline(std::cin, file\_name);

std::cout << std::endl;

checkFileName(file\_name);

checkFileExistance(file\_name, op);

file.open(file\_name);

is\_stream\_opened = true;

}

catch (const std::exception&)

{

std::cout << "Ошибка при попытке открыть файл." << std::endl;

continue;

}

catch (int)

{

continue;

}

} while (!is\_stream\_opened);

while (true)

{

file << counter << std::endl;

try {

for (auto& elem : shp) {

if (elem->getName() == "круг") {

auto tmp = std::dynamic\_pointer\_cast<circle>(elem);

file << tmp->getName() << std::endl;

file << tmp->getRadius() << std::endl;

}

else if (elem->getName() == "прямоугольник") {

auto tmp = std::dynamic\_pointer\_cast<rectangle>(elem);

file << tmp->getName() << std::endl;

file << tmp->getHeight() << std::endl;

file << tmp->getWidht() << std::endl;

}

else {

auto tmp = std::dynamic\_pointer\_cast<trapezoid>(elem);

file << tmp->getName() << std::endl;

file << tmp->getLargeBase() << std::endl;

file << tmp->getLeftSide() << std::endl;

file << tmp->getRightSide() << std::endl;

file << tmp->getSmallerBase() << std::endl;

}

}

file.close();

}

catch (const std::exception&)

{

std::cout << "Ошибка при записи информации в файл. Повторите ввод." << std::endl;

continue;

}

std::cout << "Данные успешно сохранены." << std::endl;

break;

}

}

void fileInput(std::vector<std::shared\_ptr<figure>>& shp)

{

std::ifstream file;

file.exceptions(std::istream::badbit | std::ifstream::failbit);

bool is\_stream\_opened = false;

std::string file\_name = "";

do

{

try

{

std::cout << "Введите полное имя файла (название и тип, пример: название.txt) или путь до него: ";

std::getline(std::cin, file\_name);

checkFileName(file\_name);

checkFileExistance(file\_name, ip);

file.open(file\_name);

is\_stream\_opened = true;

}

catch (const std::exception&)

{

continue;

}

} while (!is\_stream\_opened);

while (true)

{

try

{

int size = getLineI(file);

for (int i = 0; i < size; i++)

{

std::string name = getLineS(file);

if (name == "круг")

{

auto c\_ptr = std::make\_shared<circle>();

c\_ptr->setRadius(getLineD(file));

shp.push\_back(c\_ptr);

}

else if (name == "прямоугольник")

{

auto r\_ptr = std::make\_shared<rectangle>();

r\_ptr->setHeight(getLineD(file));

r\_ptr->setWidth(getLineD(file));

shp.push\_back(r\_ptr);

}

else if (name == "трапеция")

{

auto t\_ptr = std::make\_shared<trapezoid>();

t\_ptr->setLargeBase(getLineD(file));

t\_ptr->setLeftSide(getLineD(file));

t\_ptr->setRightSide(getLineD(file));

t\_ptr->setSmallerBase(getLineD(file));

if (t\_ptr->getSmallerBase() > t\_ptr->getLargeBase()) throw 1;

shp.push\_back(t\_ptr);

}

else throw 1;

}

}

catch (int)

{

continue;

}

catch (const std::exception&)

{

std::cout << "Ошибка при чтении информации из файла. Повторите ввод." << std::endl;

}

break;

}

}

void fileOutputFin(std::vector<std::shared\_ptr<figure>>& shp)

{

int index = 1;

std::ofstream file;

file.exceptions(std::ofstream::badbit | std::ofstream::failbit);

bool is\_stream\_opened = false;

std::string file\_name = "";

do

{

try

{

std::cout << "Введите имя полное имя файла (название и тип, пример: название.txt) или путь до него: ";

std::getline(std::cin, file\_name);

std::cout << std::endl;

checkFileName(file\_name);

checkFileExistance(file\_name, op);

file.open(file\_name);

is\_stream\_opened = true;

}

catch (const std::exception&)

{

continue;

}

catch (int)

{

continue;

}

} while (!is\_stream\_opened);

while (true)

{

try

{

for (auto&& elem : shp) {

file << "Объект № " << index << " Название фигуры: ";

file << elem->getName() << std::endl;

file << "Площадь: " << elem->getSquare() << std::endl;

file << "Периметр: " << elem->getPerimeter() << std::endl;

file << "" << std::endl;

index++;

}

std::cout << "Данные успешно записаны." << std::endl;

file.close();

break;

}

catch (const std::exception&)

{

std::cout << "Ошибка при записи информации в файл." << std::endl;

}

}

}

6)Interface.h

#pragma once

#include <iostream>

void showGreetings();

void showTask();

void showInputChoice();

void showFigureMenu();

void showOutputMenu();

void showContMenu();

void showFinOutputMenu();

void showEndMenu();

6)Interface.cpp

#include "interface.h"

void showGreetings(void)

{

std::cout << "Контрольная работа №2, вариант 5, сделано студентом группы 415 Шарашидзе Н.Л." << std::endl;

}

void showTask(void)

{

std::cout << "Создать базовый класс фигура, производные классы: круг, прямоугольник, трапеция." << std::endl;

std::cout << "Определить виртуальные функции для вычисления площади и периметра." << std::endl;

}

void showInputChoice(void)

{

std::cout << "Выберете способ ввода данных:" << std::endl;

std::cout << "0 - Завершить работу" << std::endl;

std::cout << "1 - Из файла" << std::endl;

std::cout << "2 - На консоль" << std::endl;

std::cout << "Ввод: ";

}

void showFigureMenu(void)

{

std::cout << "Выберете, что хотите сделать:" << std::endl;

std::cout << "0 - Вернуться к предыдущиму выбору" << std::endl;

std::cout << "1 - Круг" << std::endl;

std::cout << "2 - Прямоугольник" << std::endl;

std::cout << "3 - Трапеция" << std::endl;

std::cout << "Ввод: ";

}

void showOutputMenu(void)

{

std::cout << "Вы хотите записать данные в исходном виде в файл?" << std::endl;

std::cout << "0 - Нет" << std::endl;

std::cout << "1 - Да" << std::endl;

std::cout << "Ввод: ";

}

void showContMenu(void)

{

std::cout << "Хотите ввести новую фигуру?" << std::endl;

std::cout << "0 - Нет" << std::endl;

std::cout << "1 - Да" << std::endl;

std::cout << "Ввод: ";

}

void showFinOutputMenu(void)

{

std::cout << "Хотите записать данные в красиво офрмленном виде в новый файл" << std::endl;

std::cout << "0 - Нет" << std::endl;

std::cout << "1 - Да" << std::endl;

std::cout << "Ввод: ";

}

void showEndMenu(void)

{

std::cout << "Вы хотите выполнить программу снова?" << std::endl;

std::cout << "0 - Нет" << std::endl;

std::cout << "1 - Да" << std::endl;

std::cout << "Ввод: ";

}

7)Program.cpp

#include "program.h"

int program(void)

{

SetConsoleCP (1251); //Установлена кодировка ввода

SetConsoleOutputCP (1251); //Установлена кодировка вывода на консооль

setlocale (LC\_CTYPE, "RU"); //Установка русской локали

showGreetings();

showTask();

//Заполнение массива фигур

while(true)

{

int user\_choice = yes, counter = 0;

bool is\_con\_input = false;

std::vector<std::shared\_ptr<figure>> shp;

while (user\_choice == yes) {

shp.clear();

shp.resize(0);

showInputChoice();

user\_choice = getStartChoice();

if (user\_choice == quit) exit(0);

else if (user\_choice == console) {

is\_con\_input = true;

do

{

showFigureMenu();

user\_choice = getFigureChoice();

if (user\_choice == \_circle) { circleChoice(shp); counter++; }

else if (user\_choice == \_rectangle) { rectangleChoice(shp); counter++; }

else if (user\_choice == \_trapezoid) { trapezoidChoice(shp); counter++; }

else

{

user\_choice = yes;

break;

}

showContMenu();

user\_choice = getBinChoice();

} while (user\_choice != quit);

}

else fileInput(shp);

}

if (is\_con\_input)

{

showOutputMenu();

user\_choice = getBinChoice();

if (user\_choice == yes) { fileOutput(shp, counter); }

}

showResult(shp);

showFinOutputMenu();

user\_choice = getBinChoice();

if(user\_choice == yes) fileOutputFin(shp);

showEndMenu();

user\_choice = getBinChoice();

if (user\_choice == no) break;

};

return 0;

}