**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и  программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 7**

Тема: Проектирование структуры классов

Студент: Юревич В.Ю.

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. **Постановка задачи**

Спроектировать простейший «графический» векторный редактор.

Требование к функционалу редактора:

* создание нового документа
* импорт документа из файла
* экспорт документа в файл
* создание графического примитива (согласно варианту задания)
* удаление графического примитива
* отображение документа на экране (печать перечня графических объектов и их характеристик в std::cout)
* реализовать операцию undo, отменяющую последнее сделанное действие. (должно действовать для операций добавления/удаления фигур.)

Вариант 13.

Фигуры по варианту: ромб, пятиугольник, шестиугольник.

**Ссылка на репозиторий на GitHub:** https://github.com/vi-yurevich/oop\_exercise\_07

1. **Описание программы**

В программе реализован класс TFigure и его наследники - TRhombus, TPentagone, THexagone. В каждом классе есть конструкторы, определяющие координаты вершин для каждой фигуры, а также метод получения вектора координат, получения стороны фигуры и ее имени. Все эти методы реализованы аналогично 3-й лабораторной работе. Класс  TFactory, реализован с помощью применения паттерна проектирования Singleton, позволяющего использовать единственный экземпляр класса для создания фигур. Одним из атрибутов класса TOrg является стек векторов, содержащий себе предыдущие состояния. Метод этого класса - SaveState() - добавляет вектор на вершину стека, тем самым фиксируя состояние данных перед каждым действием пользователя. Метод Undo() - выдает элемент, который лежит на вершине стека и удаляет его с вершины, таким образом возвращая состояние данных до совершения действия. Имеется возможность экспорта текущего состояния в файл и импорта состояния из уже имеющегося файла. Реализован тщательный контроль вводимых данных.

1. **Руководство по использованию программы**

Взаимодействие с пользователем происходит с помощью меню:

1 - Вывод меню

2 - Импорт данных

3 - Экспорт текущего состояния в файл

4 - Добавить ромб

5 - Добавить пятиугольник

6 - Добавить шестиугольник

7 - Удалить элемент по индексу

8 - Печать элементов на экран

9 - Отмена последнего действия

10 - Очистка

0 - Выход из программы

1. **Набор  тестов**

test\_01.txt:

4 //добавляем ромб

-1

-2

-5 //пытаемся ввести некорректные данные

a

-10b

10

12

5 //добавляем пятиугольник

1

1

2

8 //выводим элементы на экран

7 //удаляем элемент по индексу

-1 //попытка указать некорректный индекс

1

8 //выводим элементы на экран

9 //возвращаем предыдущее состояние

8 //выводим элементы на экран

0 //выход из программы

test\_02.txt:

6 //добавляем шестиугольник

1

1

1

6 //добавляем шестиугольник

5

-1

2

3 //экспортируем данные в файл

f //вводим имя файла

10 //очищаем текущее состояние

8 //попытка напечатать имеющиеся фигуры

2 //импортируем данные

f //указываем имя файла

8 //печатаем текущее состояние

0 //выход

1. **Результаты выполнения тестов**

test\_01.txt:

./oop\_exercise\_07

1 - Вывод меню

2 - Импорт данных

3 - Экспорт текущего состояния в файл

4 - Добавить ромб

5 - Добавить пятиугольник

6 - Добавить шестиугольник

7 - Удалить элемент по индексу

8 - Печать элементов на экран

9 - Отмена последнего действия

10 - Очистка

0 - Выход из программы

Выберете пункт меню: 4

Укажите координату Х центра фигуры: -1

Укажите координату Y центра фигуры: -2

Укажите длину вертикальной диагонали: -5

Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.

Укажите длину вертикальной диагонали: a

Ошибка ввода. Повторите ввод.

Укажите длину вертикальной диагонали: -10b

Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.

Укажите длину вертикальной диагонали: Ошибка ввода. Повторите ввод.

Укажите длину вертикальной диагонали: 10

Укажите длину горизонтальной диагонали: 12

Выберете пункт меню: 5

Укажите координату Х центра фигуры: 1

Укажите координату Y центра фигуры: 1

Укажите длину стороны: 2

Выберете пункт меню: 8

[0] Тип: Ромб

Координаты точек: (-1 ; 3) (5 ; -2) (-1 ; -7) (-7 ; -2)

Координаты центра: (-1 ; -2)

Площадь: 60

[1] Тип: Пятиугольник

Координаты точек: (1 ; 2.7013) (2.7013 ; 1) (2 ; -0.701302) (0 ; -0.701302) (-0.701302 ; 1)

Координаты центра: (1 ; 1)

Площадь: 6.88191

Выберете пункт меню: 7

Укажите индекс: -1

Ошибка. Элемент с указанным индексом отсутствует в массиве. Повторите ввод.

Укажите индекс: 1

Выберете пункт меню: 8

[0] Тип: Пятиугольник

Координаты точек: (1 ; 2.7013) (2.7013 ; 1) (2 ; -0.701302) (0 ; -0.701302) (-0.701302 ; 1)

Координаты центра: (1 ; 1)

Площадь: 6.88191

Выберете пункт меню: 9

Выберете пункт меню: 8

[0] Тип: Ромб

Координаты точек: (-1 ; 3) (5 ; -2) (-1 ; -7) (-7 ; -2)

Координаты центра: (-1 ; -2)

Площадь: 60

[1] Тип: Пятиугольник

Координаты точек: (1 ; 2.7013) (2.7013 ; 1) (2 ; -0.701302) (0 ; -0.701302) (-0.701302 ; 1)

Координаты центра: (1 ; 1)

Площадь: 6.88191

Выберете пункт меню: 0

test\_02.txt:

1 - Вывод меню

2 - Импорт данных

3 - Экспорт текущего состояния в файл

4 - Добавить ромб

5 - Добавить пятиугольник

6 - Добавить шестиугольник

7 - Удалить элемент по индексу

8 - Печать элементов на экран

9 - Отмена последнего действия

10 - Очистка

0 - Выход из программы

Выберете пункт меню: 6

Укажите координату Х центра фигуры: 1

Укажите координату Y центра фигуры: 1

Укажите длину стороны: 1

Выберете пункт меню: 6

Укажите координату Х центра фигуры: 5

Укажите координату Y центра фигуры: -1

Укажите длину стороны: 2

Выберете пункт меню: 3

Введите имя файла: f

Выберете пункт меню: 10

Выберете пункт меню: 8

Сначала нужно ввести хотя бы одну фигуру

Выберете пункт меню: 2

Введите имя файла: f

Выберете пункт меню: 8

[0] Тип: Шестиугольник

Координаты точек: (0.5 ; 1.86603) (1.5 ; 1.86603) (2 ; 1) (1.5 ; 0.133975) (0.5 ; 0.133975) (0 ; 1)

Координаты центра: (1 ; 1)

Площадь: 2.59808

[1] Тип: Шестиугольник

Координаты точек: (4 ; 0.732051) (6 ; 0.732051) (7 ; -1) (6 ; -2.73205) (4 ; -2.73205) (3 ; -1)

Координаты центра: (5 ; -1)

Площадь: 10.3923

Выберете пункт меню: 0

1. **Листинг программы**

**main.cpp:**

#include <iostream>

#include <memory>

#include <stack>

#include <limits>

#include "Figure.h"

class TFactory {

public:

   static TFactory& Object() {

       static TFactory ob;

       return ob;

   }

   std::shared\_ptr<TFigure> Rhomb(double x0, double y0, double h, double v) {

       return std::make\_shared<TRhombus>(x0, y0, h, v);

   }

   std::shared\_ptr<TFigure> Pent(double x0, double y0, double s) {

       return std::make\_shared<TPentagone>(x0, y0, s);

   }

   std::shared\_ptr<TFigure> Hex(double x0, double y0 , double s) {

       return std::make\_shared<THexagone>(x0, y0, s);

   }

private:

   TFactory() {}

   TFactory(const TFactory& ob) = delete;

   TFactory& operator=(const TFactory&) = delete;

};

struct memento {

   std::vector<std::shared\_ptr<TFigure> > state;

   memento() {}

   memento(const std::vector<std::shared\_ptr<TFigure>> &other) : state({other}) {}

   ~memento() {}

};

struct TOrg {

   std::stack<memento> tmp;

   void SaveState(std::vector<std::shared\_ptr<TFigure>>  nextmove) {

       tmp.emplace(nextmove);

   }

   std::vector<std::shared\_ptr<TFigure>> Undo() {

       if (!tmp.empty()) {

           std::vector<std::shared\_ptr<TFigure>> result = tmp.top().state;

           tmp.pop();

           return result;

       } else {

           throw std::logic\_error("Ошибка. Отсутсвуют предыдущие состояния\n");

       }

   }

};

std::ostream& operator<<(std::ostream &os, std::vector<std::shared\_ptr<TFigure>>& vec) {

   if (vec.empty() == false ) {

       for (int j = 0; j < vec.size(); ++j) {

                   std::cout << "[" << j << "] ";

                   std::cout << "Тип: " << vec[j]->Name() << std::endl;

                   std::cout << "Координаты точек: " << vec[j]->Coordinate() << std::endl;

                   std::cout << "Координаты центра: " << vec[j]->Center() << std::endl;

                   std::cout << "Площадь: " << vec[j]->Area() << std::endl;

       }

   }

   else {

       std::cout << "Сначала нужно ввести хотя бы одну фигуру\n";

   }

   return os;

}

void AddFigureToVec(char type, std::vector<std::shared\_ptr<TFigure>>& vec, TOrg& act) {

   bool key = false;

   double x0, y0, side, vert, hor;

   do {

       std::cout << "Укажите координату Х центра фигуры: ";

       std::cin >> x0;

       if (std::cin.fail()) {

           std::cin.clear();

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

           std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

       }

       else {

           key = true;

           std::cin.clear();

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

       }

   } while (key != true);

   do {

       std::cout << "Укажите координату Y центра фигуры: ";

       std::cin >> y0;

       if (std::cin.fail()) {

           std::cin.clear();

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

           std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

       }

       else {

           key = false;

           std::cin.clear();

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

       }

   } while (key != false);

   switch(type) {

       case 'r': {

           do {

               std::cout << "Укажите длину вертикальной диагонали: ";

               std::cin >> vert;

               if (std::cin.fail()) {

                   std::cin.clear();

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

                   std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (vert < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = true;

                   std::cin.clear();

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

               }

           } while(key != true);

           do {

               std::cout << "Укажите длину горизонтальной диагонали: ";

               std::cin >> hor;

               if (std::cin.fail()) {

               std::cin.clear();

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

               std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (hor < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = false;

                   std::cin.clear();

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

               }

           } while(key != false);

           act.SaveState(vec);

           vec.push\_back(TFactory::Object().Rhomb(x0, y0, vert, hor));

           break;

       }

       case 'p': {

           do {

               std::cout << "Укажите длину стороны: ";

               std::cin >> side;

               if (std::cin.fail()) {

                   std::cin.clear();

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

                   std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (side < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = true;

                   std::cin.clear();

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

               }

           } while(key != true);

           act.SaveState(vec);

           vec.push\_back(TFactory::Object().Pent(x0, y0, side));

           break;

       }

       case 'h': {

           do {

               std::cout << "Укажите длину стороны: ";

               std::cin >> side;

               if (std::cin.fail()) {

                   std::cin.clear();

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

                   std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (side < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = true;

                   std::cin.clear();

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

               }

           } while(key != true);

           act.SaveState(vec);

           vec.push\_back(TFactory::Object().Hex(x0, y0, side));

           break;

       }

   }

}

int main () {

int menu;

bool key;

double x, y, side, horDiag, vertDiag;

TOrg act;

std::vector<std::shared\_ptr<TFigure>> vec;

menu = 1;

key = false;

while(menu != 0) {

   switch(menu) {

   case 1:

       std::cout << "1 - Вывод меню\n2 - Импорт данных\n3 - Экспорт текущего состояния в файл\n";

       std::cout << "4 - Добавить ромб\n5 - Добавить пятиугольник\n6 - Добавить шестиугольник\n";

       std::cout << "7 - Удалить элемент по индексу\n8 - Печать элементов на экран\n";

       std::cout << "9 - Отмена последнего действия\n10 - Очистка\n0 - Выход из программы\n";

       break;

   case 2: {

       char typeFig;

       act.SaveState(vec);

       vec.clear();

       std::string fileName;

       FILE\* file = nullptr;

       std::cout << "Введите имя файла: ";

       std::cin >> fileName;

       file = fopen(fileName.c\_str(), "r");

       if (file == nullptr) {

           std::cout << "Ошибка. Невозможно открыть файл" << std::endl;

           vec = act.Undo();

           break;

       }

       while (!feof(file)) {

           fread(&typeFig, sizeof(char), 1, file);

           if (typeFig == 'r') {

           fread(&vertDiag, sizeof(double), 1, file);

           fread(&horDiag, sizeof(double), 1, file);

           }

           else {

               fread(&side, sizeof(double), 1, file);

           }

           fread(&x, sizeof(double), 1, file);

           fread(&y, sizeof(double), 1, file);

           switch (typeFig) {

           case 'r':

               vec.push\_back(TFactory::Object().Rhomb(x, y, vertDiag, horDiag));

               break;

           case 'p':

               vec.push\_back(TFactory::Object().Pent(x, y, side));

               break;

           case 'h':

               vec.push\_back(TFactory::Object().Hex(x, y, side));

               break;

           }

       }

       vec.pop\_back();

       fclose(file);

       break;

   }

   case 3: {

       if (vec.empty() == true) {

           std::cout << "Сначала нужно ввести хотя бы одну фигуру\n";

           break;

       }

       std::string fileName;

       FILE\* file = nullptr;

       char typeFig;

       std::cout << "Введите имя файла: ";

       std::cin >> fileName;

       file = fopen(fileName.c\_str(), "w");

       if (file == NULL) {

           std::cout << "Ошибка. Невозможно открыть файл" << std::endl;

           break;

       }

       int g = 0;

       for (int j = 0; j < vec.size(); ++j) {

           if (vec[j]->Name() == "Ромб") typeFig = 'r';

           else if (vec[j]->Name() == "Пятиугольник") typeFig = 'p';

           else if (vec[j]->Name() == "Шестиугольник") typeFig = 'h';

           fwrite(&typeFig, sizeof(char), 1, file);

           switch(typeFig) {

               case 'r': {

                   vertDiag = vec[j]->Side().x;

                   horDiag = vec[j]->Side().y;

                   fwrite(&vertDiag, sizeof(double), 1, file);

                   fwrite(&horDiag, sizeof(double), 1, file);

                   break;

               }

               default: {

                   side = vec[j]->Side().x;

                   fwrite(&side, sizeof(double), 1, file);

                   break;

               }

           }

           x = vec[j]->Center().x;

           y = vec[j]->Center().y;

           fwrite(&x, sizeof(double), 1, file);

           fwrite(&y, sizeof(double), 1, file);

       }

       fclose(file);

       break;

   }

   case 4:

       AddFigureToVec('r', vec, act);

       break;

   case 5:

       AddFigureToVec('p', vec, act);

       break;

   case 6:

       AddFigureToVec('h', vec, act);

       break;

   case 7: {

       int j;

       bool key;

       key = false;

       do {

           std::cout << "Укажите индекс: ";

           std::cin >> j;

           if (std::cin.fail()) {

           std::cin.clear();

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

               std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

           }

           else if ((j < 0) || (j > (vec.size() - 1))) {

               std::cout << "Ошибка. Элемент с указанным индексом отсутствует в массиве. Повторите ввод.\n";

           }

           else {

               key = true;

               std::cin.clear();

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

           }

       } while(key != true);

       act.SaveState(vec);

       j = (j == 0) ? j : --j;

       vec.erase(vec.begin() + j);

       break;

   }

   case 8: {

       std::cout << vec;

       break;

   }

   case 9: {

       try {

           vec = act.Undo();

           break;

       }

       catch(std::logic\_error& error){

           std::cout << error.what();

       }

   }

   case 10:

   act.SaveState(vec);

   vec.clear();

   break;

}

do {

   std::cout << "Выберете пункт меню: ";

   std::cin >> menu;

   if (std::cin.fail()) {

       std::cin.clear();

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

       std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

   }

   else if ((menu < 0) || (menu > 11)) {

       std::cout << "Указанный пункт отсутствует в меню. Попробуйте ещё раз." << std::endl;

   }

   else {

       key = true;

       std::cin.clear();

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

   }

} while(key != true);

key = false;

}

return(0);

}

**Figure..h:**

#include <iostream>

#include <memory>

#include <stack>

#include <limits>

#include "Figure.h"

class TFactory {

public:

   static TFactory& Object() {

       static TFactory ob;

       return ob;

   }

   std::shared\_ptr<TFigure> Rhomb(double x0, double y0, double h, double v) {

       return std::make\_shared<TRhombus>(x0, y0, h, v);

   }

   std::shared\_ptr<TFigure> Pent(double x0, double y0, double s) {

       return std::make\_shared<TPentagone>(x0, y0, s);

   }

   std::shared\_ptr<TFigure> Hex(double x0, double y0 , double s) {

       return std::make\_shared<THexagone>(x0, y0, s);

   }

private:

   TFactory() {}

   TFactory(const TFactory& ob) = delete;

   TFactory& operator=(const TFactory&) = delete;

};

struct memento {

   std::vector<std::shared\_ptr<TFigure> > state;

   memento() {}

   memento(const std::vector<std::shared\_ptr<TFigure>> &other) : state({other}) {}

   ~memento() {}

};

struct TOrg {

   std::stack<memento> tmp;

   void SaveState(std::vector<std::shared\_ptr<TFigure>>  nextmove) {

       tmp.emplace(nextmove);

   }

   std::vector<std::shared\_ptr<TFigure>> Undo() {

       if (!tmp.empty()) {

           std::vector<std::shared\_ptr<TFigure>> result = tmp.top().state;

           tmp.pop();

           return result;

       } else {

           throw std::logic\_error("Ошибка. Отсутсвуют предыдущие состояния\n");

       }

   }

};

std::ostream& operator<<(std::ostream &os, std::vector<std::shared\_ptr<TFigure>>& vec) {

   if (vec.empty() == false ) {

       for (int j = 0; j < vec.size(); ++j) {

                   std::cout << "[" << j << "] ";

                   std::cout << "Тип: " << vec[j]->Name() << std::endl;

                   std::cout << "Координаты точек: " << vec[j]->Coordinate() << std::endl;

                   std::cout << "Координаты центра: " << vec[j]->Center() << std::endl;

                   std::cout << "Площадь: " << vec[j]->Area() << std::endl;

       }

   }

   else {

       std::cout << "Сначала нужно ввести хотя бы одну фигуру\n";

   }

   return os;

}

void AddFigureToVec(char type, std::vector<std::shared\_ptr<TFigure>>& vec, TOrg& act) {

   bool key = false;

   double x0, y0, side, vert, hor;

   do {

       std::cout << "Укажите координату Х центра фигуры: ";

       std::cin >> x0;

       if (std::cin.fail()) {

           std::cin.clear();

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

           std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

       }

       else {

           key = true;

           std::cin.clear();

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

       }

   } while (key != true);

   do {

       std::cout << "Укажите координату Y центра фигуры: ";

       std::cin >> y0;

       if (std::cin.fail()) {

           std::cin.clear();

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

           std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

       }

       else {

           key = false;

           std::cin.clear();

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

       }

   } while (key != false);

   switch(type) {

       case 'r': {

           do {

               std::cout << "Укажите длину вертикальной диагонали: ";

               std::cin >> vert;

               if (std::cin.fail()) {

                   std::cin.clear();

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

                   std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (vert < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = true;

                   std::cin.clear();

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

               }

           } while(key != true);

           do {

               std::cout << "Укажите длину горизонтальной диагонали: ";

               std::cin >> hor;

               if (std::cin.fail()) {

               std::cin.clear();

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

               std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (hor < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = false;

                   std::cin.clear();

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

               }

           } while(key != false);

           act.SaveState(vec);

           vec.push\_back(TFactory::Object().Rhomb(x0, y0, vert, hor));

           break;

       }

       case 'p': {

           do {

               std::cout << "Укажите длину стороны: ";

               std::cin >> side;

               if (std::cin.fail()) {

                   std::cin.clear();

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

                   std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (side < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = true;

                   std::cin.clear();

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

               }

           } while(key != true);

           act.SaveState(vec);

           vec.push\_back(TFactory::Object().Pent(x0, y0, side));

           break;

       }

       case 'h': {

           do {

               std::cout << "Укажите длину стороны: ";

               std::cin >> side;

               if (std::cin.fail()) {

                   std::cin.clear();

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

                   std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

               }

               else if (side < 0) {

                           std::cout << "Ошибка ввода. Длина не может быть отрицательной. Повторите ввод.\n";

                       }

               else {

                   key = true;

                   std::cin.clear();

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

               }

           } while(key != true);

           act.SaveState(vec);

           vec.push\_back(TFactory::Object().Hex(x0, y0, side));

           break;

       }

   }

}

int main () {

int menu;

bool key;

double x, y, side, horDiag, vertDiag;

TOrg act;

std::vector<std::shared\_ptr<TFigure>> vec;

menu = 1;

key = false;

while(menu != 0) {

   switch(menu) {

   case 1:

       std::cout << "1 - Вывод меню\n2 - Импорт данных\n3 - Экспорт текущего состояния в файл\n";

       std::cout << "4 - Добавить ромб\n5 - Добавить пятиугольник\n6 - Добавить шестиугольник\n";

       std::cout << "7 - Удалить элемент по индексу\n8 - Печать элементов на экран\n";

       std::cout << "9 - Отмена последнего действия\n10 - Очистка\n0 - Выход из программы\n";

       break;

   case 2: {

       char typeFig;

       act.SaveState(vec);

       vec.clear();

       std::string fileName;

       FILE\* file = nullptr;

       std::cout << "Введите имя файла: ";

       std::cin >> fileName;

       file = fopen(fileName.c\_str(), "r");

       if (file == nullptr) {

           std::cout << "Ошибка. Невозможно открыть файл" << std::endl;

           vec = act.Undo();

           break;

       }

       while (!feof(file)) {

           fread(&typeFig, sizeof(char), 1, file);

           if (typeFig == 'r') {

           fread(&vertDiag, sizeof(double), 1, file);

           fread(&horDiag, sizeof(double), 1, file);

           }

           else {

               fread(&side, sizeof(double), 1, file);

           }

           fread(&x, sizeof(double), 1, file);

           fread(&y, sizeof(double), 1, file);

           switch (typeFig) {

           case 'r':

               vec.push\_back(TFactory::Object().Rhomb(x, y, vertDiag, horDiag));

               break;

           case 'p':

               vec.push\_back(TFactory::Object().Pent(x, y, side));

               break;

           case 'h':

               vec.push\_back(TFactory::Object().Hex(x, y, side));

               break;

           }

       }

       vec.pop\_back();

       fclose(file);

       break;

   }

   case 3: {

       if (vec.empty() == true) {

           std::cout << "Сначала нужно ввести хотя бы одну фигуру\n";

           break;

       }

       std::string fileName;

       FILE\* file = nullptr;

       char typeFig;

       std::cout << "Введите имя файла: ";

       std::cin >> fileName;

       file = fopen(fileName.c\_str(), "w");

       if (file == NULL) {

           std::cout << "Ошибка. Невозможно открыть файл" << std::endl;

           break;

       }

       int g = 0;

       for (int j = 0; j < vec.size(); ++j) {

           if (vec[j]->Name() == "Ромб") typeFig = 'r';

           else if (vec[j]->Name() == "Пятиугольник") typeFig = 'p';

           else if (vec[j]->Name() == "Шестиугольник") typeFig = 'h';

           fwrite(&typeFig, sizeof(char), 1, file);

           switch(typeFig) {

               case 'r': {

                   vertDiag = vec[j]->Side().x;

                   horDiag = vec[j]->Side().y;

                   fwrite(&vertDiag, sizeof(double), 1, file);

                   fwrite(&horDiag, sizeof(double), 1, file);

                   break;

               }

               default: {

                   side = vec[j]->Side().x;

                   fwrite(&side, sizeof(double), 1, file);

                   break;

               }

           }

           x = vec[j]->Center().x;

           y = vec[j]->Center().y;

           fwrite(&x, sizeof(double), 1, file);

           fwrite(&y, sizeof(double), 1, file);

       }

       fclose(file);

       break;

   }

   case 4:

       AddFigureToVec('r', vec, act);

       break;

   case 5:

       AddFigureToVec('p', vec, act);

       break;

   case 6:

       AddFigureToVec('h', vec, act);

       break;

   case 7: {

       int j;

       bool key;

       key = false;

       do {

           std::cout << "Укажите индекс: ";

           std::cin >> j;

           if (std::cin.fail()) {

           std::cin.clear();

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

               std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

           }

           else if ((j < 0) || (j > (vec.size() - 1))) {

               std::cout << "Ошибка. Элемент с указанным индексом отсутствует в массиве. Повторите ввод.\n";

           }

           else {

               key = true;

               std::cin.clear();

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

           }

       } while(key != true);

       act.SaveState(vec);

       j = (j == 0) ? j : --j;

       vec.erase(vec.begin() + j);

       break;

   }

   case 8: {

       std::cout << vec;

       break;

   }

   case 9: {

       try {

           vec = act.Undo();

           break;

       }

       catch(std::logic\_error& error){

           std::cout << error.what();

       }

   }

   case 10:

   act.SaveState(vec);

   vec.clear();

   break;

}

do {

   std::cout << "Выберете пункт меню: ";

   std::cin >> menu;

   if (std::cin.fail()) {

       std::cin.clear();

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

       std::cout << "Ошибка ввода. Повторите ввод." << std::endl;

   }

   else if ((menu < 0) || (menu > 11)) {

       std::cout << "Указанный пункт отсутствует в меню. Попробуйте ещё раз." << std::endl;

   }

   else {

       key = true;

       std::cin.clear();

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

   }

} while(key != true);

key = false;

}

return(0);

}

**Figure.cpp:**

#include "Figure.h"

#include <cmath>

#include <vector>

#include <memory>

Pair::Pair() : x(0.0), y(0.0) {}

Pair::Pair(double a, double b) : x(a), y(b) {}

std::ostream& operator<<(std::ostream& os, const Pair& p){

   os << "(" << p.x << " ; " << p.y << ")";

   return(os);

}

std::ostream& operator<<(std::ostream& os, const std::vector<Pair>& v) {

   for(int i = 0; i < v.size(); ++i) {

       os << v[i] << " ";

   }

   return(os);

}

TFigure::TFigure(double x0, double y0) {

   center.x = x0;

   center.y = y0;

}

TFigure::TFigure() : TFigure (0.0, 0.0) {}

TRhombus::TRhombus(double x0, double y0, double v, double h) {

   center.x = x0;

   center.y = y0;

   verticalDiag = v;

   horisontalDiag = h;

   points.push\_back(Pair(x0, y0 + v/2.0));

   points.push\_back(Pair(x0 + h/2.0, y0));

   points.push\_back(Pair(x0, y0 - v/2.0));

   points.push\_back(Pair(x0 - h/2.0, y0));

}

TRhombus::TRhombus() : TFigure() {

   verticalDiag = 0.0;

   horisontalDiag = 0.0;

}

std::string TRhombus::Name() {

   return("Ромб");

}

double TRhombus::Area() {

   return (verticalDiag \* horisontalDiag / 2.0);

}

Pair TRhombus::Center() {

   return(center);

}

std::vector<Pair> TRhombus::Coordinate() {

   return(points);

}

Pair TRhombus::Side() {

   return(Pair{verticalDiag, horisontalDiag});

}

TPentagone::TPentagone(double x0, double y0, double s) {

   center.x = x0;

   center.y = y0;

   side = s;

   double r = s \* sqrt(2.0) / sqrt(5.0 - sqrt(5.0));

   points.push\_back(Pair(x0, y0 + r));

   points.push\_back(Pair(x0 + r, y0));

   points.push\_back(Pair(x0 + s / 2.0, y0 - r));

   points.push\_back(Pair(x0 - s / 2.0, y0 - r));

   points.push\_back(Pair(x0 - r, y0));

}

TPentagone::TPentagone() : TFigure() {

   side = 0.0;

}

std::string TPentagone::Name() {

   return("Пятиугольник");

}

double TPentagone::Area() {

   double a = pow(side, 2.0) \* sqrt(5.0) \* sqrt(5.0 + 2.0 \* sqrt(5.0)) / 4.0;

   return(a);

}

Pair TPentagone::Center() {

   return(center);

}

std::vector<Pair> TPentagone::Coordinate() {

   return(points);

}

Pair TPentagone::Side() {

   return(Pair{side, 0.0});

}

THexagone::THexagone(double x0, double y0, double s) {

   center.x = x0;

   center.y = y0;

   side = s;

   points.push\_back(Pair(x0 - s / 2.0, y0 + sqrt(3.0) \* s / 2.0));

   points.push\_back(Pair(x0 + s / 2.0, y0 + sqrt(3.0) \* s / 2.0));

   points.push\_back(Pair(x0 + s, y0));

   points.push\_back(Pair(x0 + s / 2.0, y0 - sqrt(3.0) \* s / 2.0));

   points.push\_back(Pair(x0 - s / 2.0, y0 - sqrt(3.0) \* s / 2.0));

   points.push\_back(Pair(x0 - s, y0));

}

THexagone::THexagone() : TFigure() {

   side = 0.0;

}

std::string THexagone::Name() {

   return("Шестиугольник");

}

double THexagone::Area() {

   double a = 3.0 \* sqrt(3.0) \* pow(side, 2.0) / 2.0;

   return(a);

}

Pair THexagone::Center() {

   return(center);

}

std::vector<Pair> THexagone::Coordinate() {

   return(points);

}

Pair THexagone::Side() {

   return(Pair{side, 0.0});

}

1. **Выводы**

    В процессе выполнения данной работы получен опыт реализации простейшего “графического” векторного редактора, с сохранением текущего состояния перед каждым действием и возможностью возврата к предыдущему состоянию.

**Список литературы**

1. Презентация “Проектируем структуру классов. ЛЕКЦИЯ 12” - Дзюба Д.В. (дата обращения: 12.12.20).
2. [cppreference.com](https://en.cppreference.com/) [Электронный ресурс]. URL: <https://en.cppreference.com/w/> (дата обращения: 12.12.20).