Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 7

Тема: Проектирование структуры классов

Студент: Павлова К.А.
Группа: М8О-306Б-18
Преподаватель:
Дата:
Оценка:

Москва, 2021

1. Постановка задачи

Вариант 16:

16.	8-угольник	Треугольник	Квадрат

2. Описание программы

Класс Document отвечает за хранение и работу с фигурами. Он позволяет добавлять, удалять фигуры, а также отменять добавление/удаление. Реализована возможность сохранять в файл/загружать из файла фигуры. Для конкретной фигуры или для всех фигур сразу можно вызывать функцию. Класс Factory содержит статическую функцию генерации фигуры по параметрам. Весь интерфейс работы с документом хранится в main.

3. Набор testcases

№	Описание	Ввод
1	Демонстрация функций работы с фигурами и сериализации	5 0 2 100 100 20 0 5 1 3 200 200 10 0 9 -1 6 1 9 -1 7 9 -1 7 9 -1 5 1 1 300 300 30 0 9 -1 4 1.txt 9 -1 2 9 -1 3 1.txt

		9 -1 0

4. Результаты выполнения тестов.

```
1. Show commands
2. New document
3. Import
4. Export
5. Add figure
6. Delete figure
7. Undo
8. Center point
9. Print points
10. Size of figure
11. Total size
12. Size less than
0. Exit
> 5
Index to insert figure at (0 - 0): 0
Choose type:
1 - Octagon
2 - Square
3 - Triangle
Type: 2
Coordinates of center: 100 100
Radius: 20
Angle: 0
> 5
Index to insert figure at (0 - 1): 1
Choose type:
1 - Octagon
2 - Square
3 - Triangle
Type: 3
Coordinates of center: 200 200
Radius: 10
Angle: 0
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
1: (210, 200), (195, 208.66), (195, 191.34)
> 6
Index: 1
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
> 7
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
1: (210, 200), (195, 208.66), (195, 191.34)
> 7
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
> 5
Index to insert figure at (0 - 1): 1
Choose type:
1 - Octagon
2 - Square
3 - Triangle
Type: 1
Coordinates of center: 300 300
```

```
Radius: 30
Angle: 0
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
1: (330, 300), (321.213, 321.213), (300, 330), (278.787, 321.213), (270, 300), (278.787, 278.787), (300, 270), (321.213, 278.787)
> 4
Filename: 1.txt
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
1: (330, 300), (321.213, 321.213), (300, 330), (278.787, 321.213), (270, 300), (278.787, 278.787), (300, 270), (321.213, 278.787)
> 2
> 9
Index (-1 to call for all figures): -1
> 3
Filename: 1.txt
> 9
Index (-1 to call for all figures): -1
0: (120, 100), (100, 120), (80, 100), (100, 80)
1: (330, 300), (321.213, 321.213), (300, 330), (278.787, 321.213), (270, 300), (278.787, 278.787), (300, 270), (321.213, 278.787)
> 0
```

5. Листинг программы

```
#include <iostream>
#include <vector>
#include <cmath>
#include <memory>
#include <algorithm>
#include <list>
#include <stack>
#include <string>
#include <fstream>

#define PI 3.14159265f

//точка многоугольника
using point = std::pair<float, float>;

//родительский класс для всех фигур
class Figure
{
public:
	//вычисление геометрического центра фигуры
	point getCenter()
	{
		point center = std::make_pair(0.0f, 0.0f);
		for (point p : m_points)
		{
			center.first += p.first;
			center.second += p.second;
		}
		center.first /= m_points.size();
		center.second /= m_points.size();
		return center;
	}

	//вывод координат вершин фигуры
	void print()
	{
		bool comma = false; //печатать запятую перед точкой или нет
		for (point p : m_points)
		{
```

```
                        if (comma) std::cout << ", ";
                        comma = true;

                        std::cout << "(" << p.first << ", " << p.second << ")";
                }
        }

        //вычисление площади
        float size()
        {
                float S = 0.0f;
                for (int i = 0; i < m_points.size() - 1; i++)
                        S += triag(m_points[0], m_points[i], m_points[i + 1]);
                return S;
        }

        //чтение из потока
        void read(std::istream& input)
        {
                for (point& p : m_points)
                        input >> p.first >> p.second;
        }
        //запись в поток
        void write(std::ostream& output)
        {
                output << m_points.size() << std::endl;
                for (point& p : m_points)
                        output << p.first << " " << p.second << std::endl;
        }

protected:
        //точки многоугольника
        std::vector<point> m_points;

        //площадь треугольника по координатам вершин
        //S = 1/2 * abs(det(x1 - x3, y1 - y3; x2 - x3, y2 - y3))
        float triag(point& a, point& b, point& c)
        {
                return 0.5f * abs((a.first - c.first) * (b.second - c.second) -
                        (b.first - c.first) * (a.second - c.second));
        }
};

//Любую фигуру вращения можно задать координатами центра, радиусом описанной окружности и углом
        поворота
//8-угольник
class Octagon : public Figure
{
public:
        //заполняем вектор вершин
        Octagon(float x, float y, float r, float a)
        {
                for (int i = 0; i < 8; i++)
                {
                        float phi = a + i * PI / 4.0f;
                        m_points.push_back(std::make_pair(r * cosf(phi) + x, r * sinf(phi) + y));
                }
        }
};
//Квадрат
class Square : public Figure
{
public:
        //заполняем вектор вершин
        Square(float x, float y, float r, float a)
        {
                for (int i = 0; i < 4; i++)
```

```
		{
			float phi = a + i * PI / 2.0f;
			m_points.push_back(std::make_pair(r * cosf(phi) + x, r * sinf(phi) + y));
		}
	}
};
//Треугольник
class Triangle : public Figure
{
public:
	//заполняем вектор вершин
	Triangle(float x, float y, float r, float a)
	{
		for (int i = 0; i < 3; i++)
		{
			float phi = a + i * PI / 1.5f;
			m_points.push_back(std::make_pair(r * cosf(phi) + x, r * sinf(phi) + y));
		}
	}
};

//генератор фигур
class Factory
{
public:
	enum class Type : int
	{
		T_None,
		T_Octagon,
		T_Square,
		T_Triangle
	};

	static std::shared_ptr<Figure> Make(Type type, float x, float y, float r, float a)
	{
		switch (type)
		{
		case Factory::Type::T_Octagon:
			return std::make_shared<Figure>(Octagon(x, y, r, a));

		case Factory::Type::T_Square:
			return std::make_shared<Figure>(Square(x, y, r, a));

		case Factory::Type::T_Triangle:
			return std::make_shared<Figure>(Triangle(x, y, r, a));
		}
		return std::shared_ptr<Figure>(nullptr);
	}

	static std::shared_ptr<Figure> Make(std::istream& input)
	{
		std::shared_ptr<Figure> fig = std::shared_ptr<Figure>(nullptr);

		int num;
		input >> num;

		switch (num)
		{
		case 3:
			fig = std::make_shared<Figure>(Triangle(0, 0, 0, 0));
			fig->read(input);
			break;

		case 4:
			fig = std::make_shared<Figure>(Square(0, 0, 0, 0));
			fig->read(input);
			break;
```

```
		case 8:
			fig = std::make_shared<Figure>(Octagon(0, 0, 0, 0));
			fig->read(input);
			break;
		}
		return fig;
	}
};

//документ
class Document
{
public:
	Document() {}
	~Document() {}

	//загрузка из файла
	Document(std::string fname)
	{
		std::ifstream file(fname);
		if (!file.good())
		{
			std::cout << "Unable to open file" << std::endl;
			return;
		}

		//число фигур
		int num;
		file >> num;

		//читаем num фигур
		while (num--)
			m_figures.push_back(Factory::Make(file));

		file.close();
	}

	//добавление/удаление фигур
	void AddFigure(std::shared_ptr<Figure> fig, int index)
	{
		m_undos.push(UndoData(std::shared_ptr<Figure>(nullptr), index));

		m_figures.insert(std::next(m_figures.begin(), index), fig);
	}
	void DeleteFigure(int index)
	{
		auto iter = std::next(m_figures.begin(), index);
		m_undos.push(UndoData(*iter, index));

		m_figures.erase(iter);
	}

	//отмена действия
	void Undo()
	{
		if (m_undos.empty()) return;

		UndoData data = m_undos.top();
		m_undos.pop();

		//отмена удаления фигуры
		if (data.fig)
			m_figures.insert(std::next(m_figures.begin(), data.index), data.fig);
		//отмена создания фигуры
		else
			m_figures.erase(std::next(m_figures.begin(), data.index));
```

```
}

//просмотр
void PrintCenters(int index)
{
	if (index == -1)
	{
		int i = 0;
		std::for_each(m_figures.begin(), m_figures.end(), [&i](std::shared_ptr<Figure>&
fig)
			{
				std::pair<float, float> p = fig->getCenter();
				std::cout << i << ": (" << p.first << ", " << p.second << ")" <<
std::endl;
				i++;
			});
	}
	else
	{
		std::shared_ptr<Figure> fig = *std::next(m_figures.begin(), index);
		std::pair<float, float> p = fig->getCenter();
		std::cout << index << ": (" << p.first << ", " << p.second << ")" << std::endl;
	}
}
void PrintPoints(int index)
{
	if (index == -1)
	{
		int i = 0;
		std::for_each(m_figures.begin(), m_figures.end(), [&i](std::shared_ptr<Figure>&
fig)
			{
				std::cout << i << ": ";
				fig->print();
				std::cout << std::endl;
				i++;
			});
	}
	else
	{
		std::shared_ptr<Figure> fig = *std::next(m_figures.begin(), index);
		std::cout << index << ": ";
		fig->print();
		std::cout << std::endl;
	}
}
void PrintSizes(int index)
{
	if (index == -1)
	{
		int i = 0;
		std::for_each(m_figures.begin(), m_figures.end(), [&i](std::shared_ptr<Figure>&
fig)
			{
				std::cout << i << ": " << fig->size() << std::endl;
				i++;
			});
	}
	else
	{
		std::shared_ptr<Figure> fig = *std::next(m_figures.begin(), index);
		std::cout << index << ": " << fig->size() << std::endl;
	}
}

//дополнительные функции для фигур
float TotalSize()
```

```
	{
		float total = 0.0f;
		for (auto& fig : m_figures) total += fig->size();
		return total;
	}
	int CountSmaller(float maxSz)
	{
		return std::count_if(m_figures.begin(), m_figures.end(),
			[&maxSz](std::shared_ptr<Figure>& fig)
			{
				return fig->size() < maxSz;
			});
	}

	//число фигур
	int FiguresCount() { return m_figures.size(); }

	//запись фигур в файл
	void Save(std::string fname)
	{
		std::ofstream file(fname);
		if (!file.good())
		{
			std::cout << "Unable to open file" << std::endl;
			return;
		}

		//число фигур
		file << m_figures.size() << std::endl;

		//читаем num фигур
		for (auto& fig : m_figures)
			fig->write(file);

		file.close();
	}

private:
	std::list<std::shared_ptr<Figure>> m_figures;


	struct UndoData
	{
		int index; //индекс, по которому удалена/добавлена фигура
		std::shared_ptr<Figure> fig; //nullptr -- если фигура была создана, иначе -- удалённая
	фигура

		UndoData(std::shared_ptr<Figure> f, int i) : fig(f), index(i) {}
	};
	std::stack<UndoData> m_undos;
};

//список команд
void showCommands()
{
	std::cout <<
		"1. Show commands" << std::endl <<
		"2. New document" << std::endl <<
		"3. Import" << std::endl <<
		"4. Export" << std::endl <<
		"5. Add figure" << std::endl <<
		"6. Delete figure" << std::endl <<
		"7. Undo" << std::endl <<
		"8. Center point" << std::endl <<
		"9. Print points" << std::endl <<
		"10. Size of figure" << std::endl <<
		"11. Total size" << std::endl <<
```

```
		"12. Size less than" << std::endl <<
		"0. Exit" << std::endl;
}

int main()
{
	//документ
	std::shared_ptr<Document> document = std::make_shared<Document>(Document());

	showCommands();

	//цикл программы
	bool loop = true;
	while (loop)
	{
		//читаем введённую команду
		std::cout << "> ";

		int command;
		std::cin >> command;

		switch (command)
		{
		case 0:
			loop = false;
			break;

		case 1:
			showCommands();
			break;

		case 2:
			document = std::make_shared<Document>(Document());
			break;

		case 3:
		{
			std::string fname = "";

			std::cout << "Filename: ";
			std::cin >> fname;

			document = std::make_shared<Document>(Document(fname));
			break;
		}
		case 4:
		{
			std::string fname = "";

			std::cout << "Filename: ";
			std::cin >> fname;

			document->Save(fname);
			break;
		}

		case 5:
		{
			int index, size = document->FiguresCount();
			std::cout << "Index to insert figure at (0 - " << size << "): ";
			std::cin >> index;

			if (index < 0 || index > size)
				std::cout << "Index out of bounds" << std::endl;
			else
			{
				std::cout << "Choose type:" << std::endl <<
```

```
                        "1 - Octagon" << std::endl <<
                        "2 - Square" << std::endl <<
                        "3 - Triangle" << std::endl <<
                        "Type: ";

                int type;
                std::cin >> type;

                Factory::Type ftype = Factory::Type::T_None;
                switch (type)
                {
                case 1:
                        ftype = Factory::Type::T_Octagon;
                        break;
                case 2:
                        ftype = Factory::Type::T_Square;
                        break;
                case 3:
                        ftype = Factory::Type::T_Triangle;
                        break;

                default:
                        std::cout << "Unknown type" << std::endl;
                        break;
                }
                if (ftype == Factory::Type::T_None) break;

                float x, y, r, a;
                std::cout << "Coordinates of center: ";
                std::cin >> x >> y;

                std::cout << "Radius: ";
                std::cin >> r;

                std::cout << "Angle: ";
                std::cin >> a;

                document->AddFigure(Factory::Make(ftype, x, y, r, a), index);
        }
        break;
}
case 6:
{
        std::cout << "Index: ";

        int index;
        std::cin >> index;
        if (index < 0 || index >= document->FiguresCount())
                std::cout << "Index out of bounds" << std::endl;
        else
                document->DeleteFigure(index);
        break;
}
case 7:
        document->Undo();
        break;

case 8:
{
        std::cout << "Index (-1 to call for all figures): ";

        int index;
        std::cin >> index;

        if (index < -1 || index >= document->FiguresCount())
                std::cout << "Index out of bounds" << std::endl;
        else
```

```
                        document->PrintCenters(index);
                break;
        }
        case 9:
        {
                std::cout << "Index (-1 to call for all figures): ";

                int index;
                std::cin >> index;

                if (index < -1 || index >= document->FiguresCount())
                        std::cout << "Index out of bounds" << std::endl;
                else
                        document->PrintPoints(index);
                break;
        }
        case 10:
        {
                std::cout << "Index (-1 to call for all figures): ";

                int index;
                std::cin >> index;

                if (index < -1 || index >= document->FiguresCount())
                        std::cout << "Index out of bounds" << std::endl;
                else
                        document->PrintSizes(index);
                break;
        }

        case 11:
        {
                std::cout << "Total size of all figures: " << document->TotalSize() <<
    std::endl;
                break;
        }
        case 12:
        {
                std::cout << "Maximum size: ";

                float maxSz;
                std::cin >> maxSz;

                std::cout << document->CountSmaller(maxSz) << " figures has size less than " <<
    maxSz << std::endl;
                break;
        }

        default:
                std::cout << "Unknown command" << std::endl;
                break;
        }
    }
  return 0;
}
```

6. Выводы:

Получены практические навыки в хороших практиках проектирования структуры классов приложения. Спроектирован простейший графический векторный редактор.

СПИСОК ЛИТЕРАТУРЫ

1. SOLID [электронный ресурс]. URL: https://habr.com/ru/post/348286/
2. C++ Programming: Code patterns design [электронный ресурс]. URL: https://en.wikibooks.org/wiki/C%2B%2B_Programming/Code/Design_Patterns