Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»
Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа
Дисциплина: «Объектно-ориентированное программирование»
III семестр
Задание 5: «Основы работы с коллекциями: итераторы»

Москва, 2019

1. **Тема**: Основы работы с коллекциями: итераторы

2. **Цель работы**: Изучение основ работы с коллекциями, знакомство с шаблоном проектирования «Итератор»

3. **Задание** (*вариант № 5*):
Фигура — шестиугольник. Контейнер — стек.

4. **Адрес репозитория на GitHub**
https://github.com/DragonKeker/oop_exercise_05

5. **Код программы на C++**
Lab5.cpp

```
#include <iostream>
#include <algorithm>
#include "hexagon.h"
#include "stack.h"

int main() {
	size_t N;
	float S;
	char option = '0';
	kontaineri::stack<Hexagon<int>> st;
	Hexagon<int>* he = nullptr;
	while (option != 'q') {
		std::cout << "choose option (m - menu)" << std::endl;
		std::cin >> option;
		switch (option) {
		case 'q':
			break;
		case 'm':
			std::cout << "1) push new element into stack\n"
				<< "2) insert element into chosen position\n"
				<< "3) pop element from the stack\n"
				<< "4) delete element from the chosen position\n"
				<< "5) print stack\n"
				<< "6) count elements with area less then chosen value\n"
				<< "7) print top element\n"
				<< "q) - quit" << std::endl;
			break;
		case '1': {
			try {
				he = new Hexagon<int>(std::cin);
			}
			catch (std::logic_error& err) {
				std::cout << err.what() << std::endl;
				break;
			}
			st.push(*he);
			delete he;
			break;
		}
		case '2': {
			std::cout << "enter position to insert" << std::endl;
			std::cin >> N;
			std::cout << "enter hexagon" << std::endl;
```

```
                        try {
                                he = new Hexagon<int>(std::cin);
                        }
                        catch (std::logic_error& err) {
                                std::cout << err.what() << std::endl;
                                break;
                        }
                        st.insert_by_number(N + 1, *he);
                        delete he;
                        break;
                }
                case '3': {
                        st.pop();
                        break;
                }
                case '4': {
                        std::cout << "enter position to delete" << std::endl;
                        std::cin >> N;
                        st.delete_by_number(N + 1);
                        break;
                }
                case '5': {
                        std::for_each(st.begin(), st.end(), [](Hexagon<int>& REC) {
REC.Print(std::cout); });
                        break;
                }
                case '6': {
                        std::cout << "enter max area" << std::endl;
                        std::cin >> S;
                        std::cout << std::count_if(st.begin(), st.end(), [=](Hexagon<int>& X)
{ return X.Area() < S; })
                                << std::endl;
                        break;
                }
                case '7': {
                        st.top().Print(std::cout);
                        break;
                }
                default:
                        std::cout << "Wrong. Try m - menu" << std::endl;
                        break;
                }
        }
        return 0;
}
```

point.h

```
#ifndef OOP_LAB5_POINT_H
#define OOP_LAB5_POINT_H

#include <iostream>

template<class T>
struct point {
        T x;
        T y;
};

template<class T>
std::istream& operator>>(std::istream& is, point<T>& p) {
        is >> p.x >> p.y;
        return is;
}
```

```
template<class T>
std::ostream& operator<<(std::ostream& os, point<T> p) {
	os << '(' << p.x << ' ' << p.y << ')';
	return os;
}

#endif
```

hexagon.h

```
#ifndef OOP_LAB5_HEXAGON_H
#define OOP_LAB5_HEXAGON_H

#include "point.h"
#include <cmath>

template <class T>
class Hexagon {
public:
	point<T> A, B, C, D, E, F;

	explicit Hexagon<T>(std::istream& is) {
		is >> A >> B >> C >> D >> E >> F;
	}

	Hexagon<T>() = default;

	double Area() {
		return (0.5 * abs(A.x * B.y + B.x * C.y + C.x * D.y + D.x * E.y + E.x * F.y
+ F.x * A.y
			- B.x * A.y - C.x * B.y - D.x * C.y - E.x * D.y - F.x * E.y - A.x *
F.y));
	}

	void Print(std::ostream& os) {
		os << A << " " << B << " " << C << " " << D << " " << E << " " << F <<
std::endl;
	}

	void operator<< (std::ostream& os) {
		os << A << " " << B << " " << C << " " << D << " " << E << " " << F;
	}
};

#endif
```

stack.h

```
#ifndef OOP_EXERCISE_05_STACK_H
#define OOP_EXERCISE_05_STACK_H

#include <iterator>
#include <memory>
#include <algorithm>

namespace kontaineri {


	template<class T>
	class stack {
	private:
		struct number;
```

```
	size_t size = 0;
public:
	stack() = default;

	class forward_iterator {
	public:
		using value_type = T;
		using reference = T &;
		using pointer = T *;
		using difference_type = std::ptrdiff_t;
		using iterator_category = std::forward_iterator_tag;
		explicit forward_iterator(number* ptr);
		T& operator*();
		forward_iterator& operator++();
		forward_iterator operator++(int);
		bool operator== (const forward_iterator& other) const;
		bool operator!= (const forward_iterator& other) const;
	private:
		number* it_ptr;
		friend stack;
	};

	forward_iterator begin();
	forward_iterator end();
	void push(const T& value);
	T& top();
	void pop();
	void delete_by_it(forward_iterator d_it);
	void delete_by_number(size_t N);
	void insert_by_it(forward_iterator ins_it, T& value);
	void insert_by_number(size_t N, T& value);
private:
	struct number {
		T value;
		std::unique_ptr<number> next_number = nullptr;
		forward_iterator next();
	};
	std::unique_ptr<number> first = nullptr;
};

template<class T>
typename stack<T>::forward_iterator stack<T>::begin() {
	return forward_iterator(first.get());
}

template<class T>
typename stack<T>::forward_iterator stack<T>::end() {
	return forward_iterator(nullptr);
}

template<class T>
void stack<T>::push(const T& value) {
	if (first == nullptr) {
		first = std::unique_ptr<number>(new number{ value });
	}
	else {
		auto* tmp = new number{ value };
		std::swap(tmp->next_number, first);
		first = std::move(std::unique_ptr<number>(tmp));
	}
	size++;
}

template<class T>
void stack<T>::pop() {
```

```
	if (size == 0) {
		throw std::logic_error("stack pustoi");
	}
	first = std::move(first->next_number);
	size--;
}

template<class T>
T& stack<T>::top() {
	if (size == 0) {
		throw std::logic_error("stack pustoi");
	}
	return first->value;
}

template<class T>
void stack<T>::delete_by_it(kontaineri::stack<T>::forward_iterator d_it) {
	forward_iterator i = this->begin(), end = this->end();
	if (d_it == end) throw std::logic_error("za granitsi");
	if (d_it == this->begin()) {
		this->pop();
		return;
	}
	while ((i.it_ptr != nullptr) && (i.it_ptr->next() != d_it)) {
		++i;
	}
	if (i.it_ptr == nullptr) throw std::logic_error("za granitsi");
	i.it_ptr->next_number = std::move(d_it.it_ptr->next_number);
	size--;
}

template<class T>
void stack<T>::delete_by_number(size_t N) {
	forward_iterator it = this->begin();
	for (size_t i = 1; i <= N; ++i) {
		if (i == N) break;
		++it;
	}
	this->delete_by_it(it);
}

template<class T>
void stack<T>::insert_by_it(kontaineri::stack<T>::forward_iterator ins_it, T&
value) {
	auto tmp = std::unique_ptr<number>(new number{ value });
	forward_iterator i = this->begin();
	if (ins_it == this->begin()) {
		tmp->next_number = std::move(first);
		first = std::move(tmp);
		size++;
		return;
	}
	while ((i.it_ptr != nullptr) && (i.it_ptr->next() != ins_it)) {
		++i;
	}
	if (i.it_ptr == nullptr) throw std::logic_error("za granitsi");
	tmp->next_number = std::move(i.it_ptr->next_number);
	i.it_ptr->next_number = std::move(tmp);
	size++;
}

template<class T>
void stack<T>::insert_by_number(size_t N, T& value) {
	forward_iterator it = this->begin();
	for (size_t i = 1; i <= N; ++i) {
```

```
                if (i == N) break;
                ++it;
            }
            this->insert_by_it(it, value);
        }

        template<class T>
        typename stack<T>::forward_iterator stack<T>::number::next() {
            return forward_iterator(this->next_number.get());
        }

        template<class T>
        stack<T>::forward_iterator::forward_iterator(kontaineri::stack<T>::number* ptr) {
            it_ptr = ptr;
        }

        template<class T>
        T& stack<T>::forward_iterator::operator*() {
            return this->it_ptr->value;
        }

        template<class T>
        typename stack<T>::forward_iterator& stack<T>::forward_iterator::operator++() {
            if (it_ptr == nullptr) throw std::logic_error("vne steka");
            *this = it_ptr->next();
            return *this;
        }

        template<class T>
        typename stack<T>::forward_iterator stack<T>::forward_iterator::operator++(int) {
            forward_iterator old = *this;
            ++* this;
            return old;
        }

        template<class T>
        bool stack<T>::forward_iterator::operator==(const forward_iterator& other) const {
            return it_ptr == other.it_ptr;
        }

        template<class T>
        bool stack<T>::forward_iterator::operator!=(const forward_iterator& other) const {
            return it_ptr != other.it_ptr;
        }

}
#endif
```

CMakeLists.txt

```
cmake_minimum_required (VERSION 3.5)

project(lab5)

add_executable(oop_exercise_05
  Lab5.cpp)

set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -Wall -Wextra")
```

set_target_properties(oop_exercise_05 PROPERTIES CXX_STANDART 14 CXX_STANDART_REQUIRED ON)

6. Набор testcases

test_01.txt	Ожидаемое действие
1	push (0,0)(1,1)(2,2)(3,3)(4,4)(5,5)
0 0 1 1 2 2 3 3 4 4 5 5	
1	push (1,1)(2,2)(3,3)(4,4)(5,5)(6,6)
1 1 2 2 3 3 4 4 5 5 6 6	
5	Печать стека
3	pop
5	Печать стека
q	Выход

test_02.txt	Ожидаемое действие
1	push (0,0)(1,1)(2,2)(3,3)(4,4)(5,5)
0 0 1 1 2 2 3 3 4 4 5 5	
1	push (1,1)(2,2)(3,3)(4,4)(5,5)(6,6)
1 1 2 2 3 3 4 4 5 5 6 6	
2	Вставка (6,6)(5,5)(4,4)(3,3)(2,2)(1,1) на
0	позицию 0
6 6 5 5 4 4 3 3 2 2 1 1	
5	Печать стека
6	Вывод количества элементов,
2	площадь которых < 2
7	
q	Выход

test_03.txt	Ожидаемое действие
2	Вставка (0,0)(1,1)(2,2)(3,3)(4,4)(5,5) на
0	позицию 0
0 0 1 1 2 2 3 3 4 4 5 5	
5	Печать стека
4	Удаление элемента с
0	позиции 0
5	Печать стека
q	Выход

7. Результаты выполнения тестов

test1
choose option (m - menu)

```
m
1) push new element into stack
2) insert element into chosen position
3) pop element from the stack
4) delete element from the chosen position
5) print stack
6) count elements with area less then chosen value
7) print top element
q) - quit
choose option (m - menu)
1
0 0 1 1 2 2 3 3 4 4 5 5
choose option (m - menu)
1
1 1 2 2 3 3 4 4 5 5 6 6
choose option (m - menu)
5
(1 1) (2 2) (3 3) (4 4) (5 5) (6 6)
(0 0) (1 1) (2 2) (3 3) (4 4) (5 5)
choose option (m - menu)
3
choose option (m - menu)
5
(0 0) (1 1) (2 2) (3 3) (4 4) (5 5)
choose option (m - menu)
q

C:\Users\Андрей\source\repos\lab5\Debug\lab5.exe (процесс 16764) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу…
```

test2

```
choose option (m - menu)
1
0 0 1 1 2 2 3 3 4 4 5 5
choose option (m - menu)
1
1 1 2 2 3 3 4 4 5 5 6 6
choose option (m - menu)
2
enter position to insert
```

```
0
enter hexagon
6 6 5 5 4 4 3 3 2 2 1 1
choose option (m - menu)
5
(6 6) (5 5) (4 4) (3 3) (2 2) (1 1)
(1 1) (2 2) (3 3) (4 4) (5 5) (6 6)
(0 0) (1 1) (2 2) (3 3) (4 4) (5 5)
choose option (m - menu)
6
enter max area
2
3
choose option (m - menu)
7
(6 6) (5 5) (4 4) (3 3) (2 2) (1 1)
choose option (m - menu)
q

C:\Users\Андрей\source\repos\lab5\Debug\lab5.exe (процесс 22280) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу…
```

test3

```
choose option (m - menu)
2
enter position to insert
0
enter hexagon
0 0 1 1 2 2 3 3 4 4 5 5
choose option (m - menu)
5
(0 0) (1 1) (2 2) (3 3) (4 4) (5 5)
choose option (m - menu)
4
enter position to delete
0
choose option (m - menu)
5
choose option (m - menu)
q
```

C:\Users\Андрей\source\repos\lab5\Debug\lab5.exe (процесс 18996) завершает работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Чтобы закрыть это окно, нажмите любую клавишу…

8. Объяснение результатов работы программы - вывод

Методы коллекции:
1) size — размер коллекции
2) element — описание элемента коллекции
3) first — головной элемент коллекции
4) push — добавление элемнета в стек
5) pop — удаление элекмента из стека
6) top — возвращает значение головного элемента стека
7) delete_by_it — удаление элемента по итератору
8) delete_by_number — уделение элемента по номеру
9) insert_by_it — вставка элемента по итератору
10) insert_by_number — удаление элемента по итератору
11) forward_iterator — реализация итератора forward_iterator

В ходе данной лабораторной работы были получены навыки работы с умными указателями.
Умные указатели полезны в работе с динамечскими структурами, как инструменты более удобного контроля за выделением и освобождением вот памяти.