Московский Авиационный Институт
(Национальный Исследовательский Университет)

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

Группа:
Студент: Пашкевич Андрей Романович
Преподаватель: Поповкин Александр Викторович

Москва, 2017

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

Вариант №17

ЦЕЛЬ РАБОТЫ
Целью лабораторной работы является:

- Закрепление навыков работы с классами.
- Знакомство с умными указателями.

Задание:
Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (колонка фигура 1), согласно вариантов задания (реализованную в ЛР1). Классы должны удовлетворять следующим правилам:

- Требования к классу фигуры аналогичны требованиям из лабораторной работы 1.
- Класс-контейнер должен соджержать объекты используя std:shared_ptr.
- Класс-контейнер должен иметь метод по добавлению фигуры в контейнер.
- Класс-контейнер должен иметь методы по получению фигуры из контейнера (опередеряется структурой контейнера).
- Класс-контейнер должен иметь метод по удалению фигуры из контейнера (опередеряется структурой контейнера).
- Класс-контейнер должен иметь перегруженный оператор по выводу контейнера в поток std::ostream (<<).
- Класс-контейнер должен иметь деструктор, удаляющий все элементы контейнера.
- Классы должны быть расположенны в раздельных файлах: отдельно заголовки (.h), отдельно описание методов (.cpp).

Нельзя использовать:

- Стандартные контейнеры std.
- Шаблоны (template).
- Объекты «по-значению»

Программа должна позволять:

- Вводить произвольное количество фигур и добавлять их в контейнер.
- Распечатывать содержимое контейнера.
- Удалять фигуры из контейнера.

Описание структуры классов и алгоритма работы программы:

Tree.cpp	
Tree();	Конструктор класса
Tree(std::sharedptr<TreeNode> node);	Конструктор класса с заданным узлом
void add(std::sharedptr<Figure> figure);	Добавление фигуры в дерево
Std::sharedptr<TreeNode>del(std::sharedptr<Figure> triangle);	Удаление из дерева по параметрам фигуры
bool empty();	Проверка дерева на пустоту
friendstd::ostream& operator<<(std::ostream& os, const Tree& tree);	Перегруженный оператор вывода
virtual ~Tree();	Деструктор класса
TreeNode.cpp	
TreeNode(const std::sharedptr<Figure>& figure);	Конструктор Класса
friend std::ostream& operator<<(std::ostream& os, const TreeNode& obj);	Перегруженный оператор ввода
Std::sharedptr<TreeNode>SetLeft(std::sharedptr<TreeNode> ptr)	Установка ссылки на левый узел
Std::sharedptr<TreeNode> GetLeft();	Получение ссылки на левый узел
Std::sharedptr<TreeNode> SetRight(Std::sharedptr<TreeNode> ptr);	Установка ссылки на правый узел
Std::sharedptr<TreeNode> GetRight();	Получение ссылки на правый узел
Std::sharedptr<Figure> GetFigure() const;	Получение фигуры из узла
virtual ~TreeNode();	Деструктор класса

Умный указатель — класс имитирующий интерфейс обычного указателя и добавляющий некую новую функциональность, например проверку границ при доступе или очистку памяти.

std::shared_ptr – умный указатель, с разделяемым владением объектом через его указатель. Несколько указателей shared_ptr могут владеть одним и тем же объектом; объект будет уничтожен, когда последний shared_ptr, указывающий на него, будет уничтожен или сброшен. Объект уничтожается с использованием delete-expression или с использованием пользовательской функции удаления объекта, переданной в конструктор shared_ptr.

Листинг программы:

Вывод:

В данной лабораторной работе мне представилась возможность познакомится с важным инструментом языка C++ таким как умные указатели. Умные указатели позволяют не задумываться о выделении памяти для объекта и устранить утечки памяти в программе.