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 №9

Вариант №17
Фигуры: треугольник, квадрат, прямоугольник
Контейнер 1-го уровня: бинарное дерево
Контейнер 2-го уровня: очередь

ЦЕЛЬ РАБОТЫ

Целью работы является:

- Знакомство с лямбда-выражениями

ЗАДАНИЕ

Используя структуры данных, разработанные для ЛР 6 (контейнер первого уровня и классы-фигуры) необходимо разработать:

- Контейнер второго уровня с испльзованием шаблонов
- Реализовать с помощью лямбда-выражений набор команд, совершающих операции над контейнером 1 уровня:
  - Генерация фигур со случайным значением параметров
  - Печать контейнера на экран
  - Удаление элементов со значением площади меньше определенного числа
- В контейнер второго уровня поместить цепочку команд
- Реализовать цикл, который проходит по всем командам в контейнере второго уровня и выполняет их, применяя к контейнеру первого уровня

Для создания потоков использовать механизмы:

- future
- packaged_task/async

Для обеспечения потоко-безопасности структур данных использовать:

- mutex
- lock_guard

Нельзя использовать:

- Стандартные контейнеры std.

ЛИСТИНГ ПРОГРАММЫ

В программе используются контейнер первого уровня (бинарное дерево), контейнер второго уровня (очередь) – для автоматического выполнения команд и классы фигур, разработанные для лабораторной работы №6.

main.cpp

```
#include <cstdlib>
#include <iostream>
#include <memory>
#include <future>
#include <thread>
#include <functional>
#include <random>
#include <string>

#include "figure.h"
#include "triangle.h"
#include "square.h"
#include "rectangle.h"
#include "node.h"
#include "tree.h"
#include "item.h"
#include "queue.h"

int main()
{
    Tree<Figure> tree;
    typedef std::function<void(void) > command;
    Queue<command> cmd;

    command cmd_insert = [&]()
    {
        std::cout << "Command: Create figure" << std::endl;

        std::default_random_engine generator;
        std::uniform_int_distribution<int> distrFigureType(1, 3);
        std::uniform_int_distribution<int> distrFigureParam(1, 100);
        for (int i = 0; i < 20; ++i)
        {
            int side = distrFigureParam(generator);
            switch (distrFigureType(generator))
            {
            case 1:
            {
                tree.add(std::shared_ptr<Figure>(new Triangle(side,side,side)));
                break;
            }
            case 2:
            {
                tree.add(std::shared_ptr<Figure>(new Triangle(side,side,side)));
                break;

            }
            case 3:
            {
                tree.add(std::shared_ptr<Figure>(new Triangle(side,side,side)));
                break;
            }
            }
        }
    };

    command cmd_print = [&]()
    {
        std::cout << "Command: Print tree" << std::endl;
        tree.print();
```

```
    };

    command cmd_delete = [&]()
    {
        std::cout << "Command: Delete figure" << std::endl;
        std::uniform_int_distribution<double> distrArea(1.0, 1000.0);


    };

    cmd.push(std::shared_ptr<command> (&cmd_insert, [](command*) { }));
    cmd.push(std::shared_ptr<command> (&cmd_print, [](command*) { }));
    cmd.push(std::shared_ptr<command> (&cmd_delete, [](command*) { }));
    cmd.push(std::shared_ptr<command> (&cmd_print, [](command*) { }));

    while (!cmd.empty())
    {
        std::shared_ptr<command> _cmd = cmd.front();
        cmd.pop();
        std::future<void> ft = std::async(*_cmd);
        ft.get();
    }

    return 0;
}
```

https://github.com/Andrew-Bir/MAIfaq8/tree/master/oop/LAB_09

ВЫВОДЫ

Лямбда-выражение (или просто *лямбда*) в C++11 — это удобный способ определения анонимного объекта-функции непосредственно в месте его вызова или передачи в функцию в качестве аргумента. Обычно лямбда-выражения используются для инкапсуляции нескольких строк кода, передаваемых алгоритмам или асинхронным методам.