Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Объектно-ориентированное программирование»

Студент: Севастьянов В. С.
Преподаватель: Поповкин А. В.
Группа: 08-207

Вариант: 19
Дата:
Оценка:
Подпись:

Москва, 2017

Лабораторная работа №3

1 Цель работы

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

2 Задача

Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий все три фигуры, согласно варианту задания
Классы должны удовлетворять следующим правилам:

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

Фигуры: Прямоугольник, трапеция, ромб.
Контейнер: N-дерево.

3 Описание

Умный указатель – класс (обычно шаблонный), имитирующий интерфейс обычного указателя и добавляющий некую новую функциональность, например, проверку границ при доступе или очистку памяти.
Существует 3 вида умных указателей стандартной библиотеки C++:

- unique ptr – обеспечивает, чтобы у базового указателя был только один владелец. Может быть передан новому владельцу, но не может быть скопирован или сделан общим. Заменяет auto ptr, использовать который не рекомендуется.
- shared ptr – умный указатель с подсчитанными ссылками. Используется, когда необходимо присвоить один необработанный указатель нескольким владельцам, например, когда копия указателя возвращается из контейнера, но требуется сохранить оригинал. Необработанный указатель не будет удален до тех пор, пока все владельцы shared ptr не выйдут из области или не откажутся от владения.
- weak ptr – умный указатель для особых случаев использования с shared ptr. weak ptr предоставляет доступ к объекту, который принадлежит одному или нескольким экземплярам shared ptr, но не участвует в подсчете ссылок. Используется, когда требуется отслеживать объект, но не требуется, чтобы он оставался в активном состоянии.

4 Исходный код

```
1  class TreeItem
2  {
3  public:
4    TreeItem(const std::shared_ptr<Rectangle>& rectangle, const std::shared_ptr<Rhombus
         >& rhombus, const std::shared_ptr<Trapeze>& trapeze, size_t key);
5    TreeItem(const TreeItem& orig);
6
7    std::shared_ptr<TreeItem> GetSon();
8    std::shared_ptr<TreeItem> GetSibling();
9    void SetSon(std::shared_ptr<TreeItem> son);
10   void SetSibling(std::shared_ptr<TreeItem> sibling);
11
12   size_t GetKey() const;
13
14   std::shared_ptr<Rectangle> GetRectangle() const;
15   std::shared_ptr<Trapeze> GetTrapeze() const;
16   std::shared_ptr<Rhombus> GetRhombus() const;
17   friend std::ostream& operator<<(std::ostream& os, const TreeItem& obj);
18
```

```
19    ~TreeItem();
20  private:
21    std::shared_ptr<TreeItem> son;
22    std::shared_ptr<TreeItem> sibling;
23    std::shared_ptr<Rectangle> rectangle;
24    std::shared_ptr<Trapeze> trapeze;
25    std::shared_ptr<Rhombus> rhombus;
26    size_t key;
27  };
28
29  class Tree
30  {
31  public:
32    Tree();
33    std::shared_ptr<TreeItem> insert(std::shared_ptr<TreeItem> node, size_t key);
34    void remove(size_t key);
35    void remove(std::shared_ptr<TreeItem> tree, size_t key);
36    bool empty();
37    std::shared_ptr<TreeItem> find(std::shared_ptr<TreeItem> tree, size_t key);
38    void Print();
39    void Print(std::shared_ptr<TreeItem> tree, size_t depth);
40    std::shared_ptr<Rectangle> rectangle(size_t key);
41    friend std::ostream& operator<<(std::ostream& os, const Tree& obj);
42    ~Tree();
43  private:
44    std::shared_ptr<TreeItem> root;
45  };
```

5 Выводы

В данной работе были получены навыки работы с умными указателями. В нашем случае были использованы shared_ptr. Их особенность в том, что они хранят количество ссылок на объект, и в случае если таковых не имеется, удаляются. Это весьма удобно, так как не нужно заботиться об удалении вручную.