Московский авиационный институт
(Национальный исследовательский университет)

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

Лабораторная работа № 5

Студент: Валов Вадим
Группа: 80-208
Преподаватель:
Дата:
Оценка:

Москва, 2019

1. Постановка задачи

Создать шаблон динамической коллекцию, согласно варианту задания:

1. Коллекция должна быть реализована с помощью умных указателей (std::shared_ptr, std::weak_ptr). Опционально использование std::unique_ptr;
2. В качестве параметра шаблона коллекция должна принимать тип данных;
3. Реализовать forward_iterator по коллекции;
4. Коллекция должны возвращать итераторы begin() и end();
5. Коллекция должна содержать метод вставки на позицию итератора insert(iterator);
6. Коллекция должна содержать метод удаления из позиции итератора erase(iterator);
7. При выполнении недопустимых операций (например выход аз границы коллекции или удаление не существующего элемента) необходимо генерировать исключения;
8. Итератор должен быть совместим со стандартными алгоритмами (например, std::count_if)
9. Коллекция должна содержать метод доступа: ○ Стек – pop, push, top; ○ Очередь – pop, push, top;

   ○Список, Динамический массив – доступ к элементу по оператору [];
10. Реализовать программу, которая: ○ Позволяет вводить с клавиатуры фигуры (с типом int в качестве параметра шаблона фигуры) и добавлять в коллекцию;

    ○Позволяет удалять элемент из коллекции по номеру элемента; ○ Выводит на экран введенные фигуры с помощью std::for_each; ○ Выводит на экран количество объектов, у которых площадь меньше заданной (с помощью

    std::count_if);

Создать набор шаблонов, создающих функции, реализующие:

1. Вычисление геометрического центра фигуры;
2. Вывод в стандартный поток вывода std::cout координат

вершин фигуры;

3. Вычисление площади фигуры;

Параметром шаблона должен являться тип класса фигуры (например Square<int>). Помимо самого класса фигуры, шаблонная функция должна уметь работать с tuple. Например, std::tuple<std::pair<int,int>, std::pair<int,int>, std::pair<int,int>> должен интерпретироваться как треугольник. std::tuple<std::pair<int,int>, std::pair<int,int>, std::pair<int,int>, std::pair<int,int>> - как квадрат. Каждый std::pair<int,int> - соответствует координатам вершины фигуры вращения.

Создать программу, которая позволяет:

- Вводить из стандартного ввода std::cin фигуры, согласно варианту задания (как в виде класса, так и в виде std::tuple).
- Вызывать для нее шаблонные функции (1-3).

При реализации шаблонных функций допускается использование вспомогательных шаблонов std::enable_if, std::tuple_size, std::is_same.

2. Описание программы

Программа содержит шаблоны классов Romb, forward_iterator, DynamicArray. Динамический массив, принимает фигуру, которую должен хранить. Forward iterator принимает класс на который должен указывать. Romb принимает тип данных в котором будут хранится вершины.

Romb содержит функции вычисления площади, функцию печати вершин и функцию расчета расстояния между 2 точками.

Программа хранит данные в виде Динамического массива, который позволяет обратиться к элементу по индексу. Удалить элемент по индексу. Вставить элемент по индексу.

Интерфейс взаимодействия с программой предоставляет пользователю 4 различных команд для исполнения. При вводе чисел от 1-4 пользователь выбирает команду для исполнения. Функционал программы поддерживает работу со стандартными std::for_each и std::count_if.

Набор testcases

Тестам на вход подаются команды для выполнения и вершины

Команды: 1.add 2.delete 3.Print 4.Print less then

Тест 1:

1 // Команда

1,2 2,3 3,4 4,5 // Координаты

1 //Команда

2,3 3,4 4,5 5,6 // Координаты

2 // команда

1 // Удаление

3 // Команда

4 // команда

10 // Аргумент

Тест 2:

1

2,4 3,5 4,6 7,9

1

3,3 5,4 8,5 1,6

1

1,2 2,3 3,4 4,5

1

2,3 3,4 4,5 5,6

2

3

3

4

10

Тест 3:

1

1,2 2,3 3,4 4,5

1

2,3 3,4 4,5 5,6

1

2,4 3,5 4,6 7,9

1

3,3 5,4 8,5 1,6

2

3

3

4

10

3. Результаты выполнения тестов.

Тест 1:

Enter command:

1.add

2.delete

3.Print

4.Print less then

>1

Enter 4 vertices

>1,2 2,3 3,4 4,5

Enter command:

1.add

2.delete

3.Print

4.Print less then

>1

Enter 4 vertices

>2,3 3,4 4,5 5,6

Enter command:

1.add

2.delete

3.Print

4.Print less then

>2

Enter index to delete

>1

Enter command:

1.add

2.delete

3.Print

4.Print less then

>3

<1, 2>

<2, 3>

<3, 4>

<4, 5>

<2, 3>

<3, 4>

<4, 5>

<5, 6>

Enter command:

1.add

2.delete

3.Print

4.Print less then

>4

Enter area. Programm will print Romds vertices < n

n = 10

Count = 2

Enter command:

1.add

2.delete

3.Print

4.Print less then

>-1

Exit programm...

Тест 2:

Enter command:

1.add

2.delete

3.Print

4.Print less then

>1

Enter 4 vertices

>2,4 3,5 4,6 7,9

Enter command:

1.add

2.delete

3.Print

4.Print less then

>1

Enter 4 vertices

>3,3 5,4 8,5 1,6

Enter command:

1.add

2.delete

3.Print

4.Print less then

>1

Enter 4 vertices

>1,2 2,3 3,4 4,5

Enter command:

1.add

2.delete

3.Print

4.Print less then

>1

Enter 4 vertices

>2,3 3,4 4,5 5,6

Enter command:

1.add

2.delete

3.Print

4.Print less then

>2

Enter index to delete

>3

Enter command:

1.add

2.delete

3.Print

4.Print less then

>3

<2, 4>

<3, 5>

<4, 6>

<7, 9>

<3, 3>

<5, 4>

<8, 5>

<1, 6>

<1, 2>

<2, 3>

<3, 4>

<4, 5>

<2, 3>

<3, 4>

<4, 5>

<5, 6>

Enter command:

1.add

2.delete

3.Print

4.Print less then

>4

Enter area. Programm will print Romds vertices < n

n = 10

Count = 3

Enter command:

1.add

2.delete

3.Print

4.Print less then

>-1

Exit programm...

Тест 3:

Enter command:
1.add
2.delete
3.Print
4.Print less then
>1
Enter 4 vertices
>1,2 2,3 3,4 4,5
Enter command:
1.add
2.delete
3.Print
4.Print less then
>1
Enter 4 vertices
>2,3 3,4 4,5 5,6
Enter command:
1.add
2.delete
3.Print
4.Print less then

```
>1
Enter 4 vertices
>2,4 3,5 4,6 7,9
Enter command:
 1.add
 2.delete
 3.Print
 4.Print less then
>1
Enter 4 vertices
>3,3 5,4 8,5 1,6
Enter command:
 1.add
 2.delete
 3.Print
 4.Print less then
>2
Enter index to delete
>3
Enter command:
 1.add
 2.delete
 3.Print
 4.Print less then
>3
<1, 2>
<2, 3>
<3, 4>
<4, 5>

<2, 3>
<3, 4>
<4, 5>
<5, 6>

<2, 4>
<3, 5>
```

<4, 6>
<7, 9>

<3, 3>
<5, 4>
<8, 5>
<1, 6>

Enter command:
 1.add
 2.delete
 3.Print
 4.Print less then
>4
Enter area. Programm will print Romds vertices < n
 n = 10
Count = 3
Enter command:
 1.add
 2.delete
 3.Print
 4.Print less then
>-1
Exit programm...

4. Листинг программы

```
#include <iostream>
#include <vector>
#include <algorithm>
#include "dynamicArray.h"
#include "forwardIterator.h"

int main()
{
    DynamicArray<Romb> arr;

    Romb a;
    int command = 1;
    while(command > 0)
```

```
    {
        Romb a;
        int k, n;
        std::cout << "Enter command:\n 1.add\n 2.delete\n 3.Print\n
4.Print less then\n>";
        std::cin >> command;
        switch (command)
        {
        case 1:
            std::cout << "Enter 4 vertices\n>";
            std::cin >> a;
            arr.add(a);
            break;
        case 2:
            std::cout << "Enter index to delete\n>";
            std::cin >> k;
            try
            {
                DynamicArray<Romb>::iterator it =
arr.returnIterator(k);
                arr.erase(it);
            }
            catch(int a)
            {
                if (a == OUT_OF_RANGE)std::cout << "ERROR: Out of
range\n";
                if (a == DOES_NOT_EXIST)std::cout << "ERROR: Does
not exist\n";
                if (a == ITERATOR_DONT_EXIST)std::cout << "ERROR: No
iterator in this array\n";
                if (a == TRY_TO_DELETE_EMPTY)std::cout << "ERROR:
Position is empty\n";
            }
            break;
        case 3:
            std::for_each(arr.begin(), arr.end(), [](Romb i)-
>void{i.printVertices();});
            break;
        case 4:
            std::cout << "Enter area. Programm will print Romds
vertices < n\n n = ";
            std::cin >> n;
            k = std::count_if(arr.begin(), arr.end(), [n](Romb i)
{return i.Area() < n;});
            std::cout << "Count = " << k;
            std::cout << std::endl;
            break;
        default:
```

```
            std::cout << "Exit programm...\n";
            break;
        }
    }
    return 0;
}
```

6. Вывод

Итераторы это инструмент для обходы коллекций. В зависимости от итератора мы можем получить его мощный функционал. Реализация итератора позволяет работать со своими особенными контейнерами для данных.

При добавлении поддержки стандартной библиотеки программа становится довольно универасльной