Копирующий конструктор, исключения – Домашнее задание 29

Пеганов Антон
peganoff2@mail.ru

17 июля 2020 г.

Задача 1. Копирующий конструктор

Создайте проект *polygon_project* со структурой показанной в листинге 1. Файлы *point.hpp*, *polygon.hpp*, *main.cpp* есть в каталоге *cpp/hw29/polygon_project* нашего репозитория.

Листинг 1: структура каталога *polygon_project*

```
.
├── include
│   ├── point.hpp
│   └── polygon.hpp
├── Makefile
└── source
    ├── main.cpp
    ├── point.cpp
    └── polygon.cpp
```

Классы `Point` и `Polygon` объявляются так, как показано в листинге 2. Класс `Polygon` содержит массив своих вершин в виде указателя `Point *vertices`. Конструкторы класса работают следующим образом.

1. Конструктор по умолчанию должен инициализировать `vertices` нулем.
2. Конструктор `Polygon(uint32_t n_vertices, const Point* vertices)` должен копировать содержимое `vertices` в `this->vertices`. Это дожно быть не присваивание указателей `this->vertices = vertices`, а создание новых объектов класса `Point`. В результате при изменении `vertices` не должно происходить изменения `this->vertices`.
3. Копирующий конструктор `Polygon(const Polygon&)` должен создавать независимый объект класса `Polygon`.

Листинг 2: объявления классов `Point` и `Polygon`

```
class Point
{
  double x, y;
  public:
    Point();
    Point(double, double);
    void set_x(double);
    void set_y(double);
    void print(bool = true);
};

class Polygon
{
  uint32_t n_vertices;
  Point *vertices;
  public:
    Polygon();
    Polygon(uint32_t n_vertices, const Point* vertices);
    Polygon(const Polygon&);
    Point& operator[](size_t);
    const Point& operator[](size_t) const;
    void print(bool = true);
};
```

Результат работы функции `main()` должен совпасть с содержимым листинга 3.

Листинг 3: Результат работы функции `main()`

```
First polygon:
[(1, 2), (2, 1), (0, 0)]
####################
After external array modification:
First polygon:
[(1, 2), (2, 1), (0, 0)]
vertices array:
(-1, 2)
(2, 1)
(0, 0)

Second polygon:
[(1, 2), (2, 1), (0, 0)]
after ‘p2‘ modification,
First polygon:
[(1, 2), (2, 1), (0, 0)]
Second polygon:
[(2, 4), (2, 1), (0, 0)]
####################
Subscript operator check
p2[1]: (2, 1)
```

Задача 2. Обработка исключительных ситуаций

Про генерацию и перехватывание исключений прочтите тут.
Про спецификацию исключений прочтите в главе 10.5.4 книги "Программирование на языке C++ в среде Qt Creator".
Напишите две функции, генерирующие `int` исключения:

1) `foo()`, которая допускает генерацию только `int` исключений,

2) `bar()`, которая запрещает генерацию любых исключений.

Вызовы функций `foo()` и `bar()` поместите каждую в свой блок `try` и перехватите исключения с помощью `catch`.

Задача 3. Пользовательские исключения

1. Допишите в проект `datetime_project` класс `WrongDateTimeException`, который будет генерироваться в качестве исключения, если конструкторам классов `Date` или `DateTime` переданы неправильные аргументы (например номер дня в месяце, равный нулю).

2. Додпишите в проект `fraction_project` класс `ZeroDenominatorException`, который будет генерироваться в качестве исключения, если знаменатель равен нулю.