Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №5 по курсу «ООП»

Студент: Носов Э. Л.
Преподаватель:
Группа: М8О-308Б-21
Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №5

Цель:

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

Вариант №29: Необходимо спроектировать и запрограммировать на языке C++ класс-контейнер первого уровня, содержащий одну фигуру (колонка фигура 1), согласно вариантам задания. Классы должны удовлетворять следующим правилам:

- Требования к классу фигуры аналогичны требованиям из лабораторной работы No1;
- Требования к классу контейнера аналогичны требованиям из лабораторной работы No2;
- Класс-контейнер должен содержать объекты используя std::shared_ptr<...>.

Программа должна позволять:

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

Ссылка на репозиторий: https://github.com/nsveml/UniProjects/tree/main/OOP/oop_exercise_5

1 Исходный код

Классы фигур реализованы так же как в предыдущей работе.

Класс контейнер TQueue определен в заголовочном файле queue.h, в нем указаны приватные атрибуты: целочисленные size и capacity, умный указатель типа shared_ptr на класс фигуры storage, в нем же объявлены методы класса, а именно публичные Top, Pop, Push, Empty, Length, Clear, конструкторы и деструктор, приватный метод Extend, необходимый для реаллокации памяти, и перегрузка оператора вывода. Определение всех этих функций вынесено в файл queue.cpp.

В файле main.cpp реализовано взаимодействие с пользователем, позволяющее создавать фигуры заданного класса и помещать их в контейнер TQueue.

figure.h

```
#include<iostream>

#ifndef figure_h
#define figure_h

class Figure{
    public:

    Figure();
    Figure(int a);
    virtual ~Figure();
    virtual size_t VertexesNumber();
    virtual float Area();
    virtual void Print(std::ostream& os);

    protected:

    size_t n;
    float* coords;
    const char* name = "Figure";

};

#include"figure.cpp"

#endif
```

figure.cpp

```
Figure::Figure(){
    n=0;
    coords=nullptr;
}

Figure::Figure(int a){
    n=a;
    coords = new float [2*n];
    std::cout<<"Enter coordinates in clockwise order:\n";
    for(long unsigned int i = 0; i<2*n; i++){
        std::cin>>coords[i];
    }
}

Figure::~Figure(){
```

```
    delete [] coords;
}

size_t Figure::VertexesNumber(){
    return n;
}

float Figure::Area(){
    float area = 0;
    for(long unsigned int i=0; i<2*n; i++){
        area += (i % 2) ? -coords[i]*coords[(i+1) % (2*n)] : coords[i]*coords[(i+3) % (2*n)];
    }
    area/=2;
    return (area<0)? -area : area;
}

void Figure::Print(std::ostream& os){
    os<<name<<":";
    for(long unsigned int i = 0; i<2*n; i+=2){
        os<<" ("<<coords[i]<<", "<<coords[i+1]<<')';
    }
    os<<'\n';
}
```

triangle.h

```
#include"figure.h"

class Triangle : public Figure{
    public:

    Triangle();

    Triangle(std::istream& is);

};

#include"triangle.cpp"
```

triangle.cpp

```
Triangle::Triangle(){
    name = "Triangle";
    n=3;
    coords = new float [2*n];
    for(long unsigned int i = 0; i<2*n; i++){
        coords[i]=0.0;
    }
}

Triangle::Triangle(std::istream& is){
    name = "Triangle";
    n=3;
    coords = new float[2*n];
    for(long unsigned int i=0; i<2*n; i++){
        is>>coords[i];
    }
}
```

queue.h

```
#ifndef QUEUE_H
#define QUEUE_H
```

```
3
4  class TQueue {
5    public:
6    TQueue();
7    TQueue(const TQueue& other);
8    void Push(const Triangle& polygon);
9    const Triangle Pop();
10   const Triangle& Top();
11   bool Empty();
12   size_t Length();
13   friend std::ostream& operator<<(std::ostream& os, const TQueue& queue);
14   void Clear();
15   virtual ~TQueue();
16
17   protected:
18   int size, capacity;
19   std::shared_ptr<Triangle[]> storage;
20   void extend();
21
22 };
23
24 #include"queue.cpp"
25
26 #endif
```

queue.cpp

```
1
2  TQueue::TQueue(){
3      size = 0;
4      capacity = 1;
5      storage.reset(new Triangle[1]);
6  }
7
8  TQueue::TQueue(const TQueue& other){
9      size = other.size;
10     capacity = other.capacity;
11     storage = std::make_shared<Triangle[]>(capacity);
12     for(int i = 0; i < size; i++){
13         storage[i]=other.storage[i];
14     }
15 }
16
17 void TQueue::Push(const Triangle& polygon){
18     if(capacity == size) extend();
19     storage[size]=polygon;
20     size++;
21 }
22
23 const Triangle TQueue::Pop(){
24     if(size > 0){
25         size--;
26         return(storage[size]);
27     }
28     else{
29         return Triangle();
30     }
31 }
32
33 const Triangle& TQueue::Top(){
34     return storage[size-1];
35 }
36
```

```
bool TQueue::Empty(){
    return bool(size);
}

size_t TQueue::Length(){
    return size;
}

std::ostream& operator<< (std::ostream& os, const TQueue& queue){
    os<<"Queue:\nsize: "<<queue.size<<"\ncapacity:"<< queue.capacity<<"\n"<<"=> ";
    for(int i = 0; i < queue.size; i++){
        os << queue.storage[i].Area() << ' ';
    }
    os << "=>\n";
    return os;
}

void TQueue::Clear(){
    size = 0;
    capacity = 1;
    storage.reset(new Triangle[1]);
}

TQueue::~TQueue(){
    size = 0;
    capacity = 0;
    storage.reset();
}


void TQueue::extend(){
    Triangle* tmp;
    capacity*=2;
    tmp = new Triangle[capacity];
    for(int i = 0; i < size; i++){
      tmp[i]=storage[i];
    }
    storage.reset(tmp);
  }
```

main.cpp

```
#include<iostream>
#include"triangle.h"
#include"queue.h"

int main(){
    std::cout<<"\
    u - push\n\
    o - pop\n\
    t - top\n\
    i - print\n\
    c - clear\n\
    q - quit\n";

    Triangle triangle;
    TQueue triangles;

    std::cout << Triangle();
    char a=' ';

    while(a!='q'){
        std::cout << "Enter commands:\n";
```

```
22          std::cin>>a;
23          switch(a){
24              case 'u':
25                  std::cout << "Enter coordinates in clockwise order:\n";
26                  std::cin >> triangle;
27                  triangles.Push(triangle);
28                  break;
29              case 'o':
30                  std::cout << triangles.Pop();
31                  break;
32              case 't':
33                  std::cout << triangles.Top();
34                  break;
35              case 'i':
36                  std::cout << triangles;
37                  break;
38              case 'c':
39                  triangles.Clear();
40                  std::cout << triangles;
41              case 'q':
42                  break;
43          }
44      }
45  }
```

2 Тестовые данные

```
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ cat test1
u
1 1
0 0
1 0
u
2 2
0 0
1 0
u
3 3
0 0
1 0
u
4 4
0 0
1 0
i
qnsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ cat test2
u
1 1
0 0
1 0
t
u
2 2
0 0
1 0
t
u
3 3
0 0
1 0
t
i
u
4 4
0 0
1 0
t
i
o
i
qnsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ cat test3
u
1 1
0 0
```

```
1 0
t
u
2 2
0 0
1 0
t
u
3 3
0 0
1 0
t
u
4 4
0 0
1 0
t
i
c
i
```

3 Работа программы

```
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ mkdir cm
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ cd cm
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5/cm$ cmake ../
CMake Deprecation Warning at CMakeLists.txt:1 (cmake_minimum_required):
  Compatibility with CMake < 2.8.12 will be removed from a future version of
  CMake.

  Update the VERSION argument <min> value or use a ...<max> suffix to tell
  CMake that the project does not need compatibility with older versions.


-- The C compiler identification is GNU 12.0.1
-- The CXX compiler identification is GNU 12.0.1
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5/cm
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5/cm$ make
[ 50%] Building CXX object CMakeFiles/oop_exercise_5.dir/main.cpp.o
[100%] Linking CXX executable "/mnt/d/X-Files/MAI/3 sem/OOp/labs/oop_exercise_5/oop_exercise_5"
[100%] Built target oop_exercise_5
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5/cm$ cd ..
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ ls
CMakeLists.txt  cm  headers  main.cpp  oop_exercise_5  test1  test2  test3
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ ./oop_exercise_5 < test1
    u - push
    o - pop
    t - top
    i - print
    c - clear
    q - quit
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
```

```
Enter coordinates in clockwise order:
Enter commands:
Queue:
size: 4
capacity:4
=> 0.5 1 1.5 2 =>
Enter commands:
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ ./oop_exercise_5 < test2
    u - push
    o - pop
    t - top
    i - print
    c - clear
    q - quit
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (1, 1) (0, 0) (1, 0)
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (2, 2) (0, 0) (1, 0)
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (3, 3) (0, 0) (1, 0)
Enter commands:
Queue:
size: 3
capacity:4
=> 0.5 1 1.5 =>
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (4, 4) (0, 0) (1, 0)
Enter commands:
Queue:
size: 4
capacity:4
=> 0.5 1 1.5 2 =>
Enter commands:
Figure: (4, 4) (0, 0) (1, 0)
Enter commands:
Queue:
size: 3
capacity:4
=> 0.5 1 1.5 =>
Enter commands:
nsveml@LAPTOP-LNCHG0M3:/mnt/d/X-Files/MAI/3 sem/OOP/labs/oop_exercise_5$ ./oop_exercise_5 < test3
```

```
    u - push
    o - pop
    t - top
    i - print
    c - clear
    q - quit
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (1, 1) (0, 0) (1, 0)
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (2, 2) (0, 0) (1, 0)
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (3, 3) (0, 0) (1, 0)
Enter commands:
Enter coordinates in clockwise order:
Enter commands:
Triangle: (4, 4) (0, 0) (1, 0)
Enter commands:
Queue:
size: 4
capacity:4
=> 0.5 1 1.5 2 =>
Enter commands:
Queue:
size: 0
capacity:1
=> =>
Enter commands:
Queue:
size: 0
capacity:1
=> =>
Enter commands:
```

4 Вывод

Выполнив пятую лабораторную работу по курсу «ООП», я закрепил полученные ранее знания, научился использовать умный указатель типа shared_ptr и реализовал с помощью него динамическую структуру данных.