Московский авиационный институт
(Национальный исследовательский университет)

Лабораторная работа № 4

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

Москва, 2019

1. Постановка задачи

Разработать шаблоны классов согласно варианту задания. Параметром шаблона должен являться скалярный тип данных задающий тип данных для оси координат. Классы должны иметь публичные поля. Фигуры являются фигурами вращения. Для хранения координат фигур необходимо использовать шаблон std::pair.

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.

Вариант 5: Фигуры - ромб, пятиугольник. шестиугольник

2. Описание программы

Программа шаблоны классов Romb, Fivethangle, Sixthangle. Внутри классов есть переменные вершин обозначенные std::pair<T, T>, где T - указываемый тип данных .

Также есть шаблонные функции, которые в зависимости от типа данных вершины и фигуры - вычисляют площадь фигуры и геометрический центр фигуры. Внутри функций происходит проверка на поданный тип данных и в зависимости от этого происходит обработка.

Программа не хранит предыдущие копии данных, а обрабатывает только текущий.

Интерфейс взаимодействия с программой предоставляет пользователю 4 различных команд для исполнения. При вводе чисел от 1-3 программа предложит ввести вершины для определенной фигуры, тип данных и способов хранения - tuple или класс. При любых других данных программа выходит.

3. Набор testcases

Тестам на вход подаются команды для выполнения и вершины

Команды: 1 - ввод ромба, 2 - ввод пятиугольника, 3 - ввод шестиугольника

1. int 2.double 3.float 4.long
1. class 2. tuple

Тест 1:

1 // Ввод ромба

1 // Ввод типа данных

1 // Ввод класс или тьюпл

2,1 1,2 2,3 3,2 // Вершины ромба

2 // Ввод пятиугольника

2 // Ввод типа данных

2 // Ввод класс или тьюпл

2,1 2,3 3,4 4,3 4,1 // Вершины пятиугольника

3 // Ввод шестиугольника

3 // Ввод типа данных

1 // Ввод класс или тьюпл

0,4 4,8 12,8 16,4 12,0 4,0 //Вершины шестиугольника

0 // Выход

Тест 2:

1 // Ввод ромба

1 // Ввод типа данных

2 // Ввод класс или тьюпл

1,1 2,2 3,2 2,1 // Координаты ромба

2 // Ввод пятиугольника

2 // Ввод тип данных

1 // Ввод класс или тьюпл

1,1 1,3 3,3 4,2 3,1 // Координаты пятиугольника

3 // Ввод шестиугольника

3 // Ввод тип данных

2 // Ввод класс или тьюпл

1.2,5.2 5.2,9.2 13.2,9.2 17.2,5.2 13.2,1.2 5.2,1.2 // Координаты шестиугольника

0 // Выход

Тест 3:

1 // Ввод ромба

1 // Ввод типа данных

2 // Ввод класс или тьюпл

-1,-1 -2,-2 -3,-2 -2,-1 // Координаты ромба

2 // Ввод пятиугольника

2 // Ввод тип данных

1 // Ввод класс или тьюпл

-1,-1 -1,-3 -3,-3 -4,-2 -3,-1 // Координаты пятиугольника

3 // Ввод шестиугольника

3 // Ввод тип данных

1 // Ввод класс или тьюпл

-1.2,-5.2 -5.2,-9.2 -13.2,-9.2 -17.2,-5.2 -13.2,-1.2 -5.2,-1.2 // Координаты шестиугольника

0 // Выход

4. Результаты выполнения тестов.

Тест 1:

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>1

Enter type 1. int 2.double 3.float 4.long

>1

class or tuple:

1.class

2.tuple

>1

Enter 4 vertices

>2,1 1,2 2,3 3,2

----- OutPut -----

Romb vertices:

<2, 1>

<1, 2>

<2, 3>

<3, 2>

Romb area: 2

Geometric center: <2, 2>

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>2

Enter type 1. int 2.double 3.float 4.long

>2

class or tuple:

1.class

2.tuple

>2

Enter 5 vertices

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

----- OutPut -----

Fivethangle vertices:

<2, 1>

<2, 3>

<3, 4>

<4, 3>

<4, 1>

Fivethangle area: 6.88191

Geometric center: <3, 2.4>

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>3

Enter type 1. int 2.double 3.float 4.long

>3

class or tuple:

1.class

2.tuple

>1

Enter 6 vertices

>0,4 4,8 12,8 16,4 12,0 4,0

----- OutPut -----

Sixthangle vertices:

<0, 4>

<4, 8>

<12, 8>

<16, 4>

<4, 0>

Sixthangle area: 83.1384

Geometric center: <8, 4>

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>-1

Exit programm

Тест 2:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>1

Enter type 1. int 2.double 3.float 4.long

>1

class or tuple:

1.class

2.tuple

>2

Enter 4 vertices

>1,1 2,2 3,2 2,1

----- OutPut -----

Romb vertices:

<1, 1>

<2, 2>

<3, 2>

<2, 1>

Romb area: 1

Geometric center: <2, 1>

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>2

Enter type 1. int 2.double 3.float 4.long

>2

class or tuple:

1.class

2.tuple

>1

Enter 5 vertices

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

----- OutPut -----

Fivethangle vertices:

<1, 1>

<1, 3>

<3, 3>

<4, 2>

<3, 1>

Fivethangle area: 6.88191

Geometric center: <2.4, 2>

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>3

Enter type 1. int 2.double 3.float 4.long

>3

class or tuple:

1.class

2.tuple

>2

Enter 6 vertices

>1.2,5.2 5.2,9.2 13.2,9.2 17.2,5.2 13.2,1.2 5.2,1.2

----- OutPut -----

Sixthangle vertices:

<1.2, 5.2>

<5.2, 9.2>

<13.2, 9.2>

<17.2, 5.2>

<13.2, 1.2>

<5.2, 1.2>

Sixthangle area: 64.9519

Geometric center: <9.2, 5.2>

Commands:

1. Add Romb

2. Add Fivethangle

3. Add Sixthangle

-1. Exit

>-1

Exit programm

Тест 3:

Commands:
1. Add Romb
2. Add Fivethangle
3. Add Sixthangle
-1. Exit
>1
Enter type 1. int 2.double 3.float 4.long
>1

```
class or tuple:
 1.class
 2.tuple
>2
Enter 4 vertices
>1,1 2,2 3,2 2,1
----- OutPut -----
Romb vertices:
<1, 1>
<2, 2>
<3, 2>
<2, 1>
Romb area: 1
Geometric center: <2, 1>
------------------
Commands:
 1. Add Romb
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit
>2
Enter type 1. int 2.double 3.float 4.long
>2
class or tuple:
 1.class
 2.tuple
>1
Enter 5 vertices
>1,1 1,3 3,3 4,2 3,1
----- OutPut -----
Fivethangle vertices:
<1, 1>
<1, 3>
<3, 3>
<4, 2>
<3, 1>
Fivethangle area: 6.88191
```

```
Geometric center: <2.4, 2>
------------------
Commands:
 1. Add Romb
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit
>3
Enter type 1. int 2.double 3.float 4.long
>3
class or tuple:
 1.class
 2.tuple
>2
Enter 6 vertices
>1.2,5.2 5.2,9.2 13.2,9.2 17.2,5.2 13.2,1.2 5.2,1.2
----- OutPut -----
Sixthangle vertices:
<1.2, 5.2>
<13.2, 9.2>
<17.2, 5.2>
<13.2, 1.2>
<5.2, 1.2>
Sixthangle area: 64.9519
Geometric center: <9.2, 5.2>
------------------
Commands:
 1. Add Romb
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit
>-1
Exit programm
PS D:\YaDisk\YandexDisk\МАИ\Andrew\2 курс\labs\University\OOP\Laba4>
.\oop_exercise_04.exe
Commands:
 1. Add Romb
```

```
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit
>1
Enter type 1. int 2.double 3.float 4.long
>1
class or tuple:
 1.class
 2.tuple
>2
Enter 4 vertices
>-1,-1 -2,-2 -3,-2 -2,-1
----- OutPut -----
Romb vertices:
<-1, -1>
<-2, -2>
<-3, -2>
<-2, -1>
Romb area: 1
Geometric center: <-2, -1>
------------------
Commands:
 1. Add Romb
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit
>2
Enter type 1. int 2.double 3.float 4.long
>2
class or tuple:
 1.class
 2.tuple
>1
Enter 5 vertices
>-1,-1 -1,-3 -3,-3 -4,-2 -3,-1
----- OutPut -----
Fivethangle vertices:
```

<-1, -1>
<-1, -3>
<-3, -3>
<-4, -2>
<-3, -1>
Fivethangle area: 6.88191
Geometric center: <-2.4, -2>

Commands:
 1. Add Romb
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit
>3
Enter type 1. int 2.double 3.float 4.long
>3
class or tuple:
 1.class
 2.tuple
>1
Enter 6 vertices
>-1.2,-5.2 -5.2,-9.2 -13.2,-9.2 -17.2,-5.2 -13.2,-1.2 -5.2,-1.2
----- OutPut -----
Sixthangle vertices:
<-1.2, -5.2>
<-5.2, -9.2>
<-13.2, -9.2>
<-17.2, -5.2>
<-5.2, -1.2>
Sixthangle area: 64.9519
Geometric center: <-9.2, -5.2>

Commands:
 1. Add Romb
 2. Add Fivethangle
 3. Add Sixthangle
-1. Exit

>-1

5. Листинг программы

```
// Эссаулов Андрей М80-207Б-18
// Разработать шаблоны классов согласно варианту задания.
// Параметром шаблона должен являться скалярный тип данных задающий
тип данных для оси координат.
// Классы должны иметь публичные поля. Фигуры являются фигурами
вращения.
// Для хранения координат фигур необходимо использовать шаблон
std::pair.

#include <iostream>
#include <vector>
#include <utility>
#include <cmath>
#include <tuple>

template <class T>
class Romb
{
public:
    friend std::istream& operator>> (std::istream& in, Romb& fig)
    {
        char k;
        T x, y;
        in >> x >> k >> y;
        fig.a = {x, y};
        in  >> x >> k >> y;
        fig.b = {x, y};
        in  >> x >> k >> y;
        fig.c = {x, y};
        in  >> x >> k >> y;
        fig.d = {x, y};
        return in;
    }
    std::pair<T, T> a, b , c, d, e, f;
};

template <class T>
class Fivethangle
{
public:
    friend std::istream& operator>> (std::istream& in, Fivethangle&
fig)
    {
        char k;
        T x, y;
```

```
        in  >> x >> k >> y;
        fig.a = {x, y};
        in  >> x >> k >> y;
        fig.b = {x, y};
        in  >> x >> k >> y;
        fig.c = {x, y};
        in  >> x >> k >> y;
        fig.d = {x, y};
        in  >> x >> k >> y;
        fig.e = {x, y};
        return in;
    }
    std::pair <T, T> a, b, c, d, e, f;
};

template <class T>
class Sixthangle
{
public:
    friend std::istream& operator>> (std::istream& in, Sixthangle&
fig)
    {
        char k;
        T x, y;
        in  >> x >> k >> y;
        fig.a = {x, y};
        in  >> x >> k >> y;
        fig.b = {x, y};
        in  >> x >> k >> y;
        fig.c = {x, y};
        in  >> x >> k >> y;
        fig.d = {x, y};
        in  >> x >> k >> y;
        fig.e = {x, y};
        in  >> x >> k >> y;
        fig.f = {x, y};
        return in;
    }
    std::pair <T, T> a, b, c, d, e, f;
};

template <class T>
T distanceBetween(std::pair<T, T> first, std::pair<T, T> second)
{
    T fir = (T)pow((abs(second.first - first.first)),2);
    T sec = (T)pow((abs(second.second - first.second)),2);
    return (T)sqrt(fir + sec);
}
```

```
template <typename TT ,template<class> class T>
std::pair<TT, TT> GeometricCenter(T<TT> fig)
{
    float x = fig.a.first + fig.b.first + fig.c.first + fig.d.first;
    float y = fig.a.second + fig.b.second + fig.c.second +
fig.d.second;

    if(std::is_same<T<TT>, Romb<TT>>::value)
    {
        x = x / 4;
        y = y / 4;
    }

    if(std::is_same<T<TT>, Fivethangle<TT>>::value)
    {
        x += fig.e.first;
        y += fig.e.second;
        x = x / 5;
        y = y / 5;
    }

    if(std::is_same<T<TT>, Sixthangle<TT>>::value)
    {
        x += fig.e.first + fig.f.first;
        y += fig.e.second + fig.f.second;
        x = x / 6;
        y = y / 6;
    }


    return {x,y};
}

template <typename TT, template<class> class T>
TT Area(T<TT> fig)
{
    if(std::is_same<T<TT>, Romb<TT>>::value)
    {
        float edgeFirst, edgeSecond;
        edgeFirst = distanceBetween<TT>(fig.a, fig.c);
        edgeSecond = distanceBetween<TT>(fig.b, fig.d);
        return (edgeFirst*edgeSecond / 2);
    }

    if(std::is_same<T<TT>, Fivethangle<TT>>::value)
    {
        TT edge = distanceBetween<TT>(fig.a, fig.b);
```

```
        TT mulconst = sqrt(25+10*sqrt(5))/4;
        return (mulconst * pow(edge, 2));
    }

    if(std::is_same<T<TT>, Sixthangle<TT>>::value)
    {
        TT edge = distanceBetween(fig.a, fig.b);
        TT mulconst = 3*sqrt(3)/2;
        return (mulconst * pow(edge, 2));
    }

}

template <typename TT , template<class> class T>
void printVertices(T<TT> fig)
{
    std::cout << "<" << fig.a.first << ", " << fig.a.second <<
">\n";
    std::cout << "<" << fig.b.first << ", " << fig.b.second <<
">\n";
    std::cout << "<" << fig.c.first << ", " << fig.c.second <<
">\n";
    std::cout << "<" << fig.d.first << ", " << fig.d.second <<
">\n";

    if(std::is_same<T<TT>, Fivethangle<TT>>::value) std::cout << "<"
<< fig.e.first << ", " << fig.e.second << ">\n";
    if(std::is_same<T<TT>, Sixthangle<TT>>::value) std::cout << "<"
<< fig.f.first << ", " << fig.f.second << ">\n";
}

template<class T>
void rombProcess(int ClassOrTuple)
{
    if(ClassOrTuple == 1)
    {
        Romb<T> buf;
        std::cout << "Enter 4 vertices\n>";
        std::cin >> buf;

        std::cout << "----- OutPut -----\n";
        std::cout << "Romb vertices: \n";
        printVertices<T, Romb>(buf);
        std::cout << "Romb area: " << Area<T, Romb>(buf) << "\n";
        std::cout << "Geometric center: <" << GeometricCenter<T,
Romb>(buf).first << ", " << GeometricCenter<T, Romb>(buf).second
<<">\n";
        std::cout << "------------------\n";
```

```
    }

    if(ClassOrTuple == 2)
    {
        char k;
        std::cout << "Enter 4 vertices\n>";
        std::pair<T, T> buf1;
        std::cin >> buf1.first >> k >> buf1.second;
        std::pair<T, T> buf2;
        std::cin >> buf2.first >> k >> buf2.second;
        std::pair<T, T> buf3;
        std::cin >> buf3.first >> k >> buf3.second;
        std::pair<T, T> buf4;
        std::cin >> buf4.first >> k >> buf4.second;
        std::tuple<std::pair<T,T>, std::pair<T,T>, std::pair<T,T>,
std::pair<T,T>> romb (buf1, buf2, buf3, buf4);
        std::cout << "----- OutPut -----\n";
        std::cout << "Romb vertices: \n";
        std::cout << "<" << buf1.first << ", " << buf1.second <<
">\n";
        std::cout << "<" << buf2.first << ", " << buf2.second <<
">\n";
        std::cout << "<" << buf3.first << ", " << buf3.second <<
">\n";
        std::cout << "<" << buf4.first << ", " << buf4.second <<
">\n";
        float edgeFirst, edgeSecond;
        edgeFirst = distanceBetween<T>(buf1, buf3);
        edgeSecond = distanceBetween<T>(buf2, buf4);
        std::cout << "Romb area: " <<  (edgeFirst*edgeSecond / 2) <<
std::endl;
        float x = (buf1.first + buf2.first + buf3.first +
buf4.first)/4;
        float y = (buf1.second + buf2.second + buf3.second +
buf4.second)/4;
        std::cout << "Geometric center: <" << x << ", " << y
<<">\n";
        std::cout << "------------------\n";
    }
}

template<class T>
void fivethangleProcess(int ClassOrTuple)
{
    if(ClassOrTuple == 1)
    {
        Fivethangle<T> buf;
        std::cout << "Enter 5 vertices\n>";
```

```
        std::cin >> buf;
        std::cout << "----- OutPut -----\n";
        std::cout << "Fivethangle vertices: \n";
        printVertices<T, Fivethangle>(buf);
        std::cout << "Fivethangle area: " << Area<T,
Fivethangle>(buf) << "\n";
        std::cout << "Geometric center: <" << GeometricCenter<T,
Fivethangle>(buf).first << ", " << GeometricCenter<T,
Fivethangle>(buf).second << ">\n";
        std::cout << "------------------\n";
    }

    if(ClassOrTuple == 2)
    {
        char k;
        std::cout << "Enter 5 vertices\n>";
        std::pair<T, T> buf1;
        std::cin >> buf1.first >> k >> buf1.second;
        std::pair<T, T> buf2;
        std::cin >> buf2.first >> k >> buf2.second;
        std::pair<T, T> buf3;
        std::cin >> buf3.first >> k >> buf3.second;
        std::pair<T, T> buf4;
        std::cin >> buf4.first >> k >> buf4.second;
        std::pair<T, T> buf5;
        std::cin >> buf5.first >> k >> buf5.second;
        std::tuple<std::pair<T,T>, std::pair<T,T>, std::pair<T,T>,
std::pair<T,T> , std::pair<T,T>> romb (buf1, buf2, buf3, buf4,
buf5);
        std::cout << "----- OutPut -----\n";
        std::cout << "Fivethangle vertices: \n";
        std::cout << "<" << buf1.first << ", " << buf1.second <<
">\n";
        std::cout << "<" << buf2.first << ", " << buf2.second <<
">\n";
        std::cout << "<" << buf3.first << ", " << buf3.second <<
">\n";
        std::cout << "<" << buf4.first << ", " << buf4.second <<
">\n";
        std::cout << "<" << buf5.first << ", " << buf5.second <<
">\n";
        T edge = distanceBetween<T>(buf1, buf2);
        T mulconst = sqrt(25+10*sqrt(5))/4;
        std::cout << "Fivethangle area: " <<  (mulconst * pow(edge,
2)) << "\n";
        float x = (buf1.first + buf2.first + buf3.first + buf4.first
+ buf5.first)/5;
        float y = (buf1.second + buf2.second + buf3.second +
```

```
buf4.second + buf5.second)/5;
        std::cout << "Geometric center: <" << x << ", " << y
<<">\n";
        std::cout << "------------------\n";
    }
}

template<class T>
void sixthangleProcess(int ClassOrTuple)
{
    if(ClassOrTuple == 1)
    {
        Sixthangle<T> buf;
        std::cout << "Enter 6 vertices\n>";
        std::cin >> buf;
        std::cout << "----- OutPut -----\n";
        std::cout << "Sixthangle vertices: \n";
        printVertices<T, Sixthangle>(buf);
        std::cout << "Sixthangle area: " << Area<T, Sixthangle>(buf)
<< "\n";
        std::cout << "Geometric center: <" << GeometricCenter<T,
Sixthangle>(buf).first << ", " << GeometricCenter<T,
Sixthangle>(buf).second << ">\n";
        std::cout << "------------------\n";
    }


    if(ClassOrTuple == 2)
    {
        char k;
        std::cout << "Enter 6 vertices\n>";
        std::pair<T, T> buf1;
        std::cin >> buf1.first >> k >> buf1.second;
        std::pair<T, T> buf2;
        std::cin >> buf2.first >> k >> buf2.second;
        std::pair<T, T> buf3;
        std::cin >> buf3.first >> k >> buf3.second;
        std::pair<T, T> buf4;
        std::cin >> buf4.first >> k >> buf4.second;
        std::pair<T, T> buf5;
        std::cin >> buf5.first >> k >> buf5.second;
        std::pair<T, T> buf6;
        std::cin >> buf6.first >> k >> buf6.second;
        std::tuple<std::pair<T,T>, std::pair<T,T>, std::pair<T,T>,
std::pair<T,T>, std::pair<T,T>, std::pair<T,T>> romb (buf1, buf2,
buf3, buf4, buf5, buf6);
        std::cout << "----- OutPut -----\n";
        std::cout << "Sixthangle vertices: \n";
```

```
        std::cout << "<" << buf1.first << ", " << buf1.second <<
">\n";
        std::cout << "<" << buf2.first << ", " << buf2.second <<
">\n";
        std::cout << "<" << buf3.first << ", " << buf3.second <<
">\n";
        std::cout << "<" << buf4.first << ", " << buf4.second <<
">\n";
        std::cout << "<" << buf5.first << ", " << buf5.second <<
">\n";
        std::cout << "<" << buf6.first << ", " << buf6.second <<
">\n";
        T edge = distanceBetween<T>(buf1, buf2);
        T mulconst = 3*sqrt(3)/2;;
        std::cout << "Sixthangle area: " <<  (mulconst * pow(edge,
2)) << "\n";
        float x = (buf1.first + buf2.first + buf3.first + buf4.first
+ buf5.first + buf6.first)/6;
        float y = (buf1.second + buf2.second + buf3.second +
buf4.second + buf5.second + buf6.second)/6;
        std::cout << "Geometric center: <" << x << ", " << y
<<">\n";
        std::cout << "------------------\n";
    }
}

int main()
{
    int command = 1, bufCommand = 1;
    float sum = 0.0f;

    while(command > 0){
        std::cout << "Commands:\n 1. Add Romb\n 2. Add Fivethangle\n
3. Add Sixthangle\n-1. Exit\n>";
        std::cin >> command;
        switch (command)
        {
        case 1:
            std::cout << "Enter type 1. int 2.double 3.float
4.long\n>";
            std::cin >> command;
            std::cout << "class or tuple:\n 1.class\n 2.tuple\n>";
            std::cin >> bufCommand;

            if(command == 1) rombProcess<int>(bufCommand);
            else if(command == 2) rombProcess<double>(bufCommand);
            else if(command == 3) rombProcess<float>(bufCommand);
            else if(command == 4) rombProcess<long>(bufCommand);
```

```
            else std::cout << "wrong type\n";
            break;
        case 2:
            std::cout << "Enter type 1. int 2.double 3.float
4.long\n>";
            std::cin >> command;
            std::cout << "class or tuple:\n 1.class\n 2.tuple\n>";
            std::cin >> bufCommand;

            if(command == 1) fivethangleProcess<int>(bufCommand);
            else if(command == 2)
fivethangleProcess<double>(bufCommand);
            else if(command == 3)
fivethangleProcess<float>(bufCommand);
            else if(command == 4)
fivethangleProcess<long>(bufCommand);
            else std::cout << "wrong type\n";
            break;
        case 3:
            std::cout << "Enter type 1. int 2.double 3.float
4.long\n>";
            std::cin >> command;
            std::cout << "class or tuple:\n 1.class\n 2.tuple\n>";
            std::cin >> bufCommand;

            if(command == 1) sixthangleProcess<int>(bufCommand);
            else if(command == 2)
sixthangleProcess<double>(bufCommand);
            else if(command == 3)
sixthangleProcess<float>(bufCommand);
            else if(command == 4)
sixthangleProcess<long>(bufCommand);
            else std::cout << "wrong type\n";
            break;
        default:
            std::cout << "Exit programm" << std::endl;
            break;
        }
    }
    return 0;
}
```

6. Вывод

Метапрограммирование это один из способов писать меньше кода. При правильном применении шаблонных классов можно сократить

количество написанных строчек кода и создать достаточный уровень абстракции для работы.

Кроме того при написании шаблонных функций можно использовать абстрагировать поступающие данных.