

Московский Авиационный Институт
(Национальный исследовательский Университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа
по курсу «ООП»**

**Тема:
Операторы, литералы.**

Студент:	Голубев В.С.
Группа:	М80-206Б-18
Преподаватель:	Журавлев А.А.
Вариант:	Quaternion
Оценка:	
Дата:	

Москва
2019

quaternion.hpp:

```

#include <iostream>
#ifndef LAB2_QUATERNION_HPP
#define LAB2_QUATERNION_HPP
struct quaternion {
    quaternion ();
    quaternion (double real, double x, double y, double z);
    quaternion operator+(const quaternion& a);
    quaternion operator-(const quaternion& a);
    quaternion operator*(const quaternion& b);
    quaternion operator/(const quaternion &b);
    friend std::ostream &operator<<(std::ostream &out, quaternion const &a);
    friend std::istream &operator>>(std::istream &in, quaternion &a);
private:
    double array[4];
};
quaternion operator"" _i(long double);
quaternion operator"" _j(long double);
quaternion operator"" _k(long double);
#endif //LAB2_QUATERNION_HPP

```

quaternion.cpp:

```

#include <iostream>
#include <cmath>
#include "quaternion.hpp"
quaternion::quaternion() {
    for (int i = 0; i < 4; ++i) {
        array[i] = 0;
    }
}
quaternion::quaternion (double real, double x, double y, double z) {
    array[0] = real;
    array[1] = x;
    array[2] = y;
    array[3] = z;
}
quaternion quaternion::operator+(const quaternion &a) {
    quaternion res;
    for (int i = 0; i < 4; ++i) {
        res.array[i] = array[i] + a.array[i];
    }
    return res;
}
quaternion quaternion::operator-(const quaternion &a) {

```

```

    quaternion res;
    for (int i = 0; i < 4; ++i) {
        res.array[i] = array[i] - a.array[i];
    }
    return res;
}

quaternion quaternion::operator*(const quaternion &b) {
    quaternion res;
    res.array[0] = array[0] * b.array[0] - array[1] * b.array[1] - array[2] * b.array[2] -
array[3] * b.array[3];
    res.array[1] = array[0] * b.array[1] + array[1] * b.array[0] + array[2] * b.array[3] -
array[3] * b.array[2];
    res.array[2] = array[0] * b.array[2] + array[2] * b.array[0] + array[3] * b.array[1] -
array[1] * b.array[3];
    res.array[3] = array[0] * b.array[3] + array[3] * b.array[0] + array[1] * b.array[2] -
array[2] * b.array[1];
    return res;
}

quaternion quaternion::operator/(const quaternion &b) {

    quaternion conjugate;
    double VectNorm;
    quaternion reverse;
    quaternion res;
    for (int i = 0; i < 4; ++i) {
        if (i == 0) {
            conjugate.array[i] = b.array[i];
        } else {
            conjugate.array[i] = -b.array[i];
        }
    }
    VectNorm = sqrt(
        b.array[0] * b.array[0] + b.array[1] * b.array[1] + b.array[2] * b.array[2] +
b.array[3] * b.array[3]);
    for (int i = 0; i < 4; ++i) {
        reverse.array[i] = conjugate.array[i] / (VectNorm * VectNorm);
    }
    res = operator*(reverse);
    return res;
}

std::ostream &operator<<(std::ostream &out, quaternion const &a) {
    out << a.array[0] << " " << "+" << " " << "(" << a.array[1] << ")" << "i" << " " <<
"+" << " " << "(" << a.array[2] << ")" << "j" << " " << "+" << "("
<< a.array[3] << ")" << "k" << std::endl;
}

```

```

    return out;
}

std::istream &operator>>(std::istream &in, quaternion &a) {
    in >> a.array[0] >> a.array[1] >> a.array[2] >> a.array[3];
    return in;
}

quaternion operator"" _i(long double i) {
    quaternion res{0, (double) i, 0, 0};
    return res;
}

quaternion operator"" _j(long double j) {
    quaternion res(0, 0, (double) j, 0);
    return res;
}

quaternion operator"" _k(long double k) {
    quaternion res(0, 0, 0, (double) k);
    return res;
}

```

main.cpp:

```

#include <iostream>
#include <cmath>
#include "quaternion.hpp"

int main() {
    quaternion q1 {}, q2 {};
    std::cin >> q1 >> q2;
    std::cout << "q1:\n" << q1 << "q2:\n" << q2;
    std::cout << "q1 + q2:\n" << q1 + q2;
    std::cout << "q1 - q2\n" << q1 - q2;
    std::cout << "q1 * q2\n" << q1 * q2;
    std::cout << "q1 / q2\n" << q1 / q2;
    quaternion a = 2.4_i;
    std::cout << a << std::endl;
    quaternion b = 3.9_j;
    std::cout << b << std::endl;
    quaternion c = 4.9_k;
    std::cout << c << std::endl;
    return 0;
}

```

CmakeLists.txt:

```
cmake_minimum_required(VERSION 3.14)
set(CMAKE_CXX_STANDARD 14)
add_executable(lab2 main.cpp
quaternion.cpp quaternion.hpp)
```

2. Ссылка на репозиторий на GitHub
[https://github.com/VSGolubev-bald /oop_exercise_02](https://github.com/VSGolubev-bald/oop_exercise_02)

3. Набор testcases.

test_00:

```
1 2 3 4
4 3 2 1
```

test_01:

```
1.5 6.2 3.46 113.3
2.67 3.2 2.98 4.01
```

test_02:

```
1.6 2.5 789.4 5.6
1.4 5.7 78.9 11.89
```

test_03:

```
2.67 3.04 -29.3 4.56
2.2 3.65 4.6 -9.03
```

test_04:

```
1.45 -0.98 2.4 5.7
1.3 4.27 -0.46 96.96
```

4. Результаты выполнения тестов.

test_00:

```
q1:
1 + (2)i + (3)j +(4)k
```

q2:
 $4 + (3)i + (2)j + (1)k$
q1 + q2:
 $5 + (5)i + (5)j + (5)k$
q1 - q2
 $-3 + (-1)i + (1)j + (3)k$
q1 * q2
 $-12 + (8)i + (24)j + (12)k$
q1 / q2
 $0.133333 + (-0.2)i + (-0.2)j + (-0.133333)k$

test_01:

q1:
 $1.5 + (6.2)i + (3.46)j + (113.3)k$
q2:
 $2.67 + (3.2)i + (2.98)j + (4.01)k$
q1 + q2:
 $4.17 + (9.4)i + (6.44)j + (117.31)k$
q1 - q2
 $-1.17 + (3)i + (0.48)j + (109.29)k$
q1 * q2
 $-480.479 + (-81.0514)i + (351.406)j + (315.93)k$
q1 / q2
 $0.0946151 + (-0.468705)i + (-0.243585)j + (-10.7333)k$

test_02:

q1:
 $1.6 + (2.5)i + (789.4)j + (5.6)k$
q2:
 $1.4 + (5.7)i + (78.9)j + (11.89)k$
q1 + q2:
 $3 + (8.2)i + (868.3)j + (17.49)k$
q1 - q2
 $0.2 + (-3.2)i + (710.5)j + (-6.29)k$
q1 * q2
 $-62362.3 + (9314.09)i + (1233.6)j + (-4275.47)k$
q1 / q2
 $0.000349944 + (-0.0022262)i + (-9.73025)j + (-0.0104021)k$

test_03:

q1:
 $2.67 + (3.04)i + (-29.3)j + (4.56)k$
q2:
 $2.2 + (3.65)i + (4.6)j + (-9.03)k$
q1 + q2:
 $4.87 + (6.69)i + (-24.7)j + (-4.47)k$
q1 - q2
 $0.47 + (-0.61)i + (-33.9)j + (13.59)k$
q1 * q2
 $170.735 + (271.852)i + (-8.0828)j + (106.851)k$
q1 / q2
 $0.0486003 + (-0.0918061)i + (1.11514)j + (0.340689)k$

test_04

```

q1:
1.45 + (-0.98)i + (2.4)j + (5.7)k
q2:
1.3 + (4.27)i + (-0.46)j + (96.96)k
q1 + q2:
2.75 + (3.29)i + (1.94)j + (102.66)k
q1 - q2
0.15 + (-5.25)i + (2.86)j + (-91.26)k
q1 * q2
-545.498 + (232.381)i + (121.813)j + (138.205)k
q1 / q2
0.000200077 + (0.00044416)i + (0.00011718)j + (-0.0586615)k

```

5. Объяснение результатов работы программы.

- 1) При запуске программы с аргументом test_?.txt объекты V1 и V2 получают компоненты q, отвечающие за действительную часть кватерниона, и компоненты x, y, z, отвечающие за мнимую векторную часть, из файлов test_?.txt.
- 2) Считывание данные для объектов при помощи перегрузки оператора ввода
- 3) Объекты выводятся на экран при помощи перегрузки оператора вывода, также все остальные выводы данных производятся при помощи данной перегрузки.
- 4) Перегружаются операторы «+» «-» «*» «/»

6. Вывод.

Выполняя данную лабораторную, я вновь поработал с простыми классами. Также изучил перегрузку операторов, осознал, насколько удобнее можно сделать свой код, работая со своими классами, как со стандартными типами данных.