# Лабораторная работа №1

## Тема: Простые классы

### Цель:

• Изучение системы сборки на языке C++, изучение систем контроля версии.

• Изучение основ работы с классами в С++;

### Порядок выполнения работы

1. Ознакомиться с теоретическим материалом.

2. Получить у преподавателя вариант задания.

3. Реализовать задание своего варианта в соответствии с поставленными требованиями.

4. Подготовить тестовые наборы данных.

5. Создать репозиторий на GitHub.

6. Отправить файлы лабораторной работы в репозиторий.

7. Отчитаться по выполненной работе путём демонстрации работающей программы на тестовых наборах

данных (как подготовленных самостоятельно, так и предложенных преподавателем) и ответов на вопросы

преподавателя (как из числа контрольных, так и по реализации программы).

### Код программы

#### main.cpp

1. #include<iostream>
2. #include "Vector3D.h"
4. #include <iomanip>
6. using namespace std;
8. int main() {
9. int x1,x2,y1,y2,z1,z2;
10. cin >> x1 >> y1 >> z1;
11. cin >> x2 >> y2 >> z2;
12. cout.precision(6);
14. Vector3D vector1(x1,y1,z1);
15. Vector3D vector2(x2,y2,z2);
16. cout << vector1.plus(vector2).getX() << " " << vector1.plus(vector2).getY() << " " << vector1.plus(vector2).getZ() << endl;
17. cout << vector1.minus(vector2).getX() << " " << vector1.minus(vector2).getY() << " " << vector1.minus(vector2).getZ() << endl;
18. cout << vector1.getAngle(vector2) << endl;
19. cout << vector1.getLength() << endl;
20. cout << vector2.getLength() << endl;
21. if (vector1.isEqual(vector2)) {
22. cout << "Equal" << endl;
23. } else {
24. cout << "Not equal" << endl;
26. }
27. cout << vector1.vectorPow(vector2).getLength() << endl;
28. return 0;
30. }

#### Vector3D.h

1. using namespace std;
3. class Vector3D
4. {
5. private:
6. double x;
7. double y;
8. double z;
10. public:
11. Vector3D(double x,double y,double z);
12. Vector3D();
13. Vector3D plus(const Vector3D& vector);
14. Vector3D minus(const Vector3D& vector);
15. Vector3D vectorPow(const Vector3D& vector);
16. void lambdaPow(double lambda);
17. double scalarPow(const Vector3D& vector);
18. bool isEqual(const Vector3D& vector);
19. double getLength();
21. double getAngle( Vector3D& vector);
23. double getX();
24. double getY();
25. double getZ();
27. };

#### Vector3D.cpp

1. #include "Vector3D.h"
3. #include <iostream>
4. #include <cmath>


8. Vector3D::Vector3D(double x,double y,double z)
9. : x(x), y(y),z(z) {}
10. Vector3D::Vector3D()
11. : x(0), y(0),z(z) {}
12. Vector3D Vector3D::plus(const Vector3D& vector) {
13. Vector3D result;
14. result.x = vector.x+this->x;
15. result.y = vector.y+this->y;
16. result.z = vector.z+this->z;
18. return result;
19. }
20. Vector3D Vector3D::minus(const Vector3D& vector) {
21. Vector3D result;
22. result.x = this->x-vector.x;
23. result.y = this->y-vector.y;
24. result.z = this->z-vector.z;
25. return result;
26. }
27. Vector3D Vector3D::vectorPow(const Vector3D& vector) {
28. Vector3D result;
29. result.x = this->y\*vector.z-this->z\*vector.y;
30. result.y = this->z\*vector.x-this->x\*vector.z;
31. result.z = this->x\*vector.y-this->y\*vector.x;
33. return result;
34. }
35. void Vector3D::lambdaPow(double lambda) {
36. this->x\*=lambda;
37. this->y\*=lambda;
38. this->z\*=lambda;
40. }
41. double Vector3D::scalarPow(const Vector3D& vector) {
42. return this->x\*vector.x+this->y\*vector.y+this->z\*vector.z;
43. }
44. bool Vector3D::isEqual(const Vector3D& vector) {
45. return (this->x==vector.x && this->y==vector.y && this->z==vector.z);
46. }
47. double Vector3D::getLength() {
48. return sqrt(this->x\*this->x+this->y\*this->y+this->z\*this->z);
49. }
51. double Vector3D::getAngle( Vector3D& vector) {
52. if ((vector.getLength()==0) || (this->getLength())==0) {
53. return 0;
54. }
55. const double halfC = 180/M\_PI;
57. double cos1 = (this->scalarPow(vector)/(this->getLength()\*vector.getLength())) ;
58. if (cos1<-1) {
59. return 180;
60. }
61. if (cos1>1) {
62. return 0;
63. }
64. return halfC\*acos(cos1);
66. }
68. double Vector3D::getX() {
69. return x;
70. }
71. double Vector3D::getY() {
72. return y;
73. }
74. double Vector3D::getZ() {
75. return z;
76. }

### Тесты:

1)

1 1 1

-1 -1 -1

2)

1 2 3

4 5 6

3)

1 1 1

1 1 1

Результаты тестирования:

1)

0 0 0

2 2 2

180

1.73205

1.73205

Not equal

0

2)

5 7 9

-3 -3 -3

12.9332

3.74166

8.77496

Not equal

7.34847

3)

2 2 2

0 0 0

0

1.73205

1.73205

Equal

0

### Объяснение результатов программы

Программа принимает на вход координаты двух векторов -- v1 и v2

На выходе программа:

1. выводит сумму v1 и v2
2. с новой строки выводит v1-v2
3. с новой строки выводит угол между v1 и v2 в градусах
4. с новой строки выводит длину v1
5. с новой строки выводит длину v2
6. с новой строки выводит Equal, если векторы равны, иначе -- Not Equal
7. с новой строки выводит длины выводит длину вектора, получившегося в результате векторного произведения