

# DSBA Introduction to Programming // Workshops 19

---

Spring semester 2023/24

Useful resource about operators: <https://en.cppreference.com/w/cpp/language/expressions>

## Exercise 1. Building the class Vector3D

---

Create a class Vector3D that represents a three-dimensional point (x,y,z) in the Cartesian space, and complete the class methods, as well as operator overloads, as indicated in each of the tasks below.

```
class Vector3D
{
public:
    /* class methods */
private:
    /* attributes */
    double _x;
    double _y;
    double _z;
};
```

### Task 1 – Empty Constructor

---

Create the empty constructor Vector3D() where every attribute is set to 0.0 ( \_x = 0.0 , \_y = 0.0 and \_z = 0.0 ).

### Task 2 – Constructor with Arguments

---

Create the constructor Vector3D(double x, double y, double z) that sets every attribute of the vector according to the arguments ( \_x = x , \_y = y , \_z = z ).

### Task 3 – Copy Constructor

---

Create the constructor Vector3D(const Vector3D& v2) that sets every attribute of the vector according to the attributes of an input vector v2 (for example, \_x = v2.\_x )

## Task 4 – getters/setters

For every attribute of the vector, write a method that returns the attribute (a getter) and a method that sets the value of the attribute according to a given argument (a setter).

For example, for attribute `_x` you should do:

```
// getter
double getX() const
{
    return x;
}

// setter
void setX(double x)
{
    _x = x;
}
```

Then, you should write a similar getter and a similar setter for attributes `_y` and `_z`.

## Task 5 – Overload the + operator

Create the overload of the operator `+` as follows: `Vector3D operator+ (const Vector3D& v1, Vector3D& v2)`, where you should return a new vector `v3` whose attributes will be the sum of `v1` and `v2`'s attributes.

For example for `v3._x`, you can implement it as: `v3.setX( v1.getX() + v2.getX() );` OR `v3.x = v1.x + v2.x;`.

## Task 6 – Overload the \* operator

Create the overload of the operator `*` as follows: `double operator* (const Vector3D& v1, Vector3D& v2)` where you should return the **dot product** between vectors `v1` and `v2`.

## Task 7 – Overload the \* operator (another version!)

Create the overload of the operator `*` as follows: `Vector3D operator* (const Vector3D& v1, double d)` where you should return a new vector `v2` whose attributes will be multiplication of every attribute of `v2` by `d`. For example, for `v2._x` you can implement it as:

```
v2.setX( v1.getX() * d );
```

## Task 8 – Vector magnitude

---

Create a member function `double magnitude()` to get the magnitude of the `Vector3D` defined as:

$$\sqrt{x^2 + y^2 + z^2}$$

## Task 9 – Overload the < operator

---

Overload the operator `<` as follows: `bool operator<(const Vector3D& v1, const Vector3D& v2)`, where:

`vector v1 < vector v2` if and only if `v1.magnitude() < v2.magnitude()`.

## Task 10 – Overload the << operator

---

Overload the print operation `<<` for a `Vector3D` to print a vector in the terminal in format `(x, y, z)`

For example, if we have a vector with attributes `_x = 3.4`, `_y = 1.0`, `_z = -1.4`, then you print in the terminal

```
(3.4, 1.0, -1.4)
```

## Task 11 – Overload the >> operator

---

Overload the read operation `>>` for a `Vector3D` to read a vector from the terminal.

For example, if the user puts in the terminal `3.4 1.0 -1.4`, then you create an object `Vector3D` with attributes `_x = 3.4`, `_y = 1.0`, `_z = -1.4`.

# Exercise 2. Using the class Vector3D

---

## Part 1

---

Create a container `std::multiset<Vector3D> s` and add 100 `Vector3D` objects to this container.

The attributes `_x`, `_y` and `_z` should be random numbers generated in the interval `[-1.0, 1.0]`.

For generating random real numbers you may use the class `std::uniform_real_distribution`

[https://www.cplusplus.com/reference/random/uniform\\_real\\_distribution/](https://www.cplusplus.com/reference/random/uniform_real_distribution/)

## Part 2

---

Print all elements stored in the container `s` using the overloaded operator.

## Part 3

---

Calculate and print the average magnitude of all vectors contained in `s`.