EIT, 306387, Gabriel Ćwiek, MAPT 2, lab 1, https://github.com/Astaree/MAPT_2

1. Ex.1: Adding “Vector” class constructors for 2d and 3d vectors by overloading. Adjust dimensions accordingly:

```
#include <iostream>
using namespace std;

class Vector
{
public:
    Vector(double);
    Vector(double, double);
    Vector(double, double, double);

    void print_v();
    size_t dimension_;

private:
    double data_[3] = {0, 0, 0};
};

Vector::Vector(double element1)
{
    cout << endl
         << "Creating a new Vector class object in R1 space:" << endl;
    dimension_ = 1;
    data_[0] = element1;
}

Vector::Vector(double element1, double element2)
{
    cout << endl
         << "Creating a new Vector class object in R2 space:" << endl;
    dimension_ = 2;
    data_[0] = element1;
    data_[1] = element2;
}

Vector::Vector(double element1, double element2, double element3)
{
    cout << endl
         << "Creating a new Vector class object in R3 space:" << endl;
    dimension_ = 3;
    data_[0] = element1;
    data_[1] = element2;
    data_[2] = element3;
}
```

```cpp
void Vector::print_v()
{
    cout << "Coords: ";
    for (auto data : data_)
    {
        cout << data << ", ";
    }
}

int main(void)
{
    Vector r1(1.0); // R1
    cout << "Vector class object with number of dimensions:"
         << r1.dimension_<<"\n";
    r1.print_v();

    Vector r2(1.0, 1.0); // R2
    cout << "Vector class object with number of dimensions:"
         << r2.dimension_ << "\n";
    r2.print_v();

    Vector r3(1.0, 1.0, 1.0); // R3
    cout << "Vector class object with number of dimensions:"
         << r3.dimension_ << "\n";
    r3.print_v();
    return EXIT_SUCCESS;
}
```

Output:

```
Creating a new Vector class object in R1 space:
Vector class object with number of dimensions:1
Coords: 1, 0, 0,
Creating a new Vector class object in R2 space:
Vector class object with number of dimensions:2
Coords: 1, 1, 0,
Creating a new Vector class object in R3 space:
Vector class object with number of dimensions:3
Coords: 1, 1, 1,
```

2. Ex. 2: Adding two overloaded functions. First one displays given value, second one displays Vector.

```
#include <iostream>
#include <iomanip>
using namespace std;

class Vector
{
public:
    Vector(double);
    Vector(double, double);
    Vector(double, double, double);

    void print_v();
    size_t dimension_;

public:
    double data_[3] = {0, 0, 0};
};

Vector::Vector(double element1)
{
    cout << endl
         << "Creating a new Vector class object in R1 space:" << endl;
    dimension_ = 1;
    data_[0] = element1;
}

Vector::Vector(double element1, double element2)
{
    cout << endl
         << "Creating a new Vector class object in R2 space:" << endl;
    dimension_ = 2;
    data_[0] = element1;
    data_[1] = element2;
}

Vector::Vector(double element1, double element2, double element3)
{
    cout << endl
         << "Creating a new Vector class object in R3 space:" << endl;
    dimension_ = 3;
    data_[0] = element1;
    data_[1] = element2;
    data_[2] = element3;
}
```

```cpp
void Vector::print_v()
{
    cout << "Coords: ";
    for (auto data : data_)
    {
        cout << data << ", ";
    }
}

void overloadedFn(double e1)
{
    cout << "Overloaded fn 1: " << setprecision(2) << e1<<endl;
}

void overloadedFn(Vector& e1)
{
    cout << "Overloaded fn 2: ";
    for (auto data : e1.data_)
    {
        cout << data << " ";
    }
    cout<<endl;
}

int main(void)
{
    Vector r1(1.0); // R1
    // cout << "Vector class object with number of dimensions:"
    //      << r1.dimension_<<"\n";
    // r1.print_v();

    Vector r2(1.0, 1.0); // R2
    // cout << "Vector class object with number of dimensions:"
    //      << r2.dimension_ << "\n";
    // r2.print_v();

    Vector r3(1.0, 1.0, 1.0); // R3
    // cout << "Vector class object with number of dimensions:"
    //      << r3.dimension_ << "\n";
    // r3.print_v();

    overloadedFn(r1.data_[0]);
    overloadedFn(r3);
    return EXIT_SUCCESS;
}
```

```
Output:

 Creating a new Vector class object in R1 space:

 Creating a new Vector class object in R2 space:

 Creating a new Vector class object in R3 space:
 Overloaded fn 1: 1

 Overloaded fn 2: 1 1 1
```

3. Ex. 3: Transform fn into methods of Vector:

```
#include <iostream>
#include <iomanip>
using namespace std;

class Vector
{
public:
    Vector(double);
    Vector(double, double);
    Vector(double, double, double);

    void overloadedFn(double e1);
    void overloadedFn(Vector &e1);
    void print_v();
    size_t dimension_;

public:
    double data_[3] = {0, 0, 0};
};

Vector::Vector(double element1)
{
    cout << endl
         << "Creating a new Vector class object in R1 space:" << endl;
    dimension_ = 1;
    data_[0] = element1;
}

Vector::Vector(double element1, double element2)
{
    cout << endl
         << "Creating a new Vector class object in R2 space:" << endl;
    dimension_ = 2;
    data_[0] = element1;
    data_[1] = element2;
}
```

```cpp
Vector::Vector(double element1, double element2, double element3)
{
    cout << endl
         << "Creating a new Vector class object in R3 space:" << endl;
    dimension_ = 3;
    data_[0] = element1;
    data_[1] = element2;
    data_[2] = element3;
}

void Vector::print_v()
{
    cout << "Coords: ";
    for (auto data : data_)
    {
        cout << data << ", ";
    }
}

void Vector::overloadedFn(double e1)
{
    cout << "Overloaded fn 1: " << setprecision(2) << e1 << endl;
}

void overloadedFn(Vector &e1)
{
    cout << "Overloaded fn 2: ";
    size_t i = 1;
    for (auto data : e1.data_)
    {
        if (i == e1.dimension_) break;
        cout << data << " ";
        i++;
    }
    cout << endl;
}

int main(void)
{
    Vector r1(1.0); // R1
    // cout << "Vector class object with number of dimensions:"
    //      << r1.dimension_<<"\n";
    // r1.print_v();

    Vector r2(1.0, 1.0); // R2
    // cout << "Vector class object with number of dimensions:"
    //      << r2.dimension_ << "\n";
    // r2.print_v();
```

```
    Vector r3(1.0, 1.0, 1.0); // R3
    // cout << "Vector class object with number of dimensions:"
    //      << r3.dimension_ << "\n";
    // r3.print_v();

    r1.overloadedFn(r1.data_[0]);
    overloadedFn(r3);
    return EXIT_SUCCESS;
}

Output:

Creating a new Vector class object in R1 space:

Creating a new Vector class object in R2 space:

Creating a new Vector class object in R3 space:
Overloaded fn 1: 1
Overloaded fn 2: 1 1
```

4. Ex. 4: Overloading addition operator to add two Vectors:

```
Vector operator+(const Vector &v1,const  Vector &v2)
{
    Vector newVector((v1.data_[0] + v2.data_[0]), (v1.data_[1] + v2.data_[1]),
(v1.data_[2] + v2.data_[2]));
    newVector.dimension_ = max(v1.dimension_, v2.dimension_);

    return newVector;

}
.
.
.
int main(void)
{
Vector r2(1.0, 1.0); // R2
Vector r3(1.0, 1.0, 1.0); // R3
Vector newV = r2+r3;
    newV.print_v();
    return EXIT_SUCCESS;
}

Output:
 Coords: 2, 2, 1,
```