1. Фигуница

```
#include <math.h>
#include <vector>
#include <string>
#include <iostream>
using namespace std;
class Figure {
protected:
    string name;
public:
    virtual void rub() = 0
    virtual void getLen() = 0
protected:
    explicit Figure(string name){
        this -> name = name;
    }
};

class Circle : public Figure {
private:
    float rad;
public:
    Circle(float r, string name):
        Figure(name), rad(r){}
    void rub() override{
        cout << "Circle: " << name << endl;
    }
```

```
    float getLen() override {
        return 2.0 * M_PI * rad;
    }
};

class Rect : public Figure {
private:
    float board;
public:
    Rect(float b, std::string name):
        Figure(name), board(b) {}
    void info() override {
        cout << "Rect: " << name << endl;
    }

    float getLen() override {
        return 4.0 * board;
    }
};
```

I. Средний

I Сергиенко

```
class Draw {
private:
    vector<Figure *> stuff;
public:
    Draw() = default;
    void addFigure(Figure *f){
        stuff.push_back(f);
    }
    float getLen(){
        float summ;
        for(auto fig : stuff){
            summ += fig->getLen();
        }
    return summ;
    }
~Draw{
    for(auto fig : stuff){
        delete fig;
    }
    stuff.clear();
}
```

Float getLen() // Периметр

```
int main(){
    Draw *d = new Draw();
    Figure* f1 = new circle(2.0, "circle");
    Figure* f2 = new Rect(10.0, "Rect");

    d -> addFigure(f1);
    d -> addFigure(f2);
    cout << d -> getLen() << endl;
    return 0;
}
```

```
#include <random>
class Vector {
private:
    int * data;
    int size;
public:
Vector(int* d=0, int s=0) {
    this->data = d;
    this->size = s;
}
Vector(int s) {
    this->size = s;
    this->data = new int[size];
    for (int i=0; i<size; ++i) {
        this->data[i] = rand()%10;
    }
Vector(const Vector & vector) {
    if (vector.size != 0) {
        delete[] this->data; this->size = vector.size;
        this->data = new int[vector.size];
        memcpy(this->data, vector.data,
        this->size * sizeof(int));
    } else {
        this->data = 0;
        this->size = 0;
    }
```

#include <string>
using namespace std;

2 Sobrecarga

```
~Vector(){
    delete[] this->data;
}

Vector& operador=(const Vector& vector){
    delete[] this->data;
    if(vector.size!=0){
        this->size=vector.size;
        this->data=new int[this->size];
        memcpy(this->data, vector.data,
        this->size*sizeof(int));
    }else{
        this->data=0;
        this->size=0;
    }
    return *this;
int getSum(){
    int sumn=0;
    for(int i=0; i<this->size;++i){
        sumn+=this->data[i];
    }
    return sumn;
}
};
```