ФГАОУВО «Южно-Уральский государственный университет (НИУ)»

Институт естественных и точных наук

Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

по дисциплине «Объектно-ориентированное программирование»

Автор работы

студент группы ЕТ-211

_____ А.А.Владимиров

_____2022 г.

Работа зачтена с оценкой

_____

_____А.К.Демидов

_____2022 г.

Челябинск, 2022

1 Постановка задачи

I. Реализовать класс

2. Вектор на плоскости (x,y) Vector

```
Конструктор: Vector(x,y)
Операции:
  a+b, a-b,
  a+=b, a-=b,
  a*b (скалярное произведение),
  a==b, a!=b
  !a (a является нулевым вектором),
  a/z, z*a ("масштабирование" вектора)
    где a,b - вектора, z - double
  вывод, ввод в виде (1.4,-2.5)
Методы:
  double getx();
  double gety();
```

Операции (если есть в задании) =, [], +=, -=, *=, /=, префиксные ++, -- определять как методы.

Для ввода переопределить >> , для вывода - << . Формат ввода-вывода объектов делать так, как указано в задании.

Запись [`текст`] означает, что `текст` может отсутствовать, например, конструктор ИмяКласса(a[,b[,c]]) может быть вызван с 1, 2 или 3 аргументами.

Пример:

```
Vector a(0,0),b(1.5,0.3);
cout<<"Введите вектор a:"<<endl;
cin>>a;
  //нужно ввести (1.2,3.2) вместе со скобками и запятой
cout<<a<<" + "<<b<<" = "<<(a+b)<<endl;
  // печатается (1.2,3.2) + (1.5,0.3) = (2.7,3.5)
```

II. Реализовать main с тестами (создание объектов и выполнение действий с ними)

2 Описание интерфейса классов

```
class Vector {
   double x, // числитель
          y; // знаменатель.
   public:
      Vector(long x=0, long y=0): x(x),
y(y){}//конструктор
      double operator*(Vector);//перегрузка операции *
      Vector &operator+=(const Vector&);//перегрузка
операции +=
      Vector &operator-=(const Vector&);//перегрузка
операции -=
      friend Vector operator*(Vector, double);
      friend Vector operator/(Vector, double);
      friend bool operator==(const Vector &, const Vector
&);//перегрузка операции ==
      friend bool operator!=(const Vector &, const Vector
&);//перегрузка операции ==
```

```
        bool operator!()const{return (x == 0 and y ==
0);}//перегрузка операции !
        friend istream &operator>>(istream &s, Vector
&p);//перегрузка операции  >>
        friend ostream &operator<<(ostream &s, const Vector
&p);// перегрузка операции  <<
        long getx() const { return x; }
        long gety() const { return y; }
  };
```

3 Описание тестов для проверки классов

```
  int main() {
     Vector a(4, 3), b(1,2);
     cout<<"Вывод\n";
     cout<<"a="<<a<<" "<<"b="<<b<<endl;
     cout<<"Ввод a\n";
     cin>>a;
     cout<<"a="<<a <<endl;

     cout<<"Проверка операции +\n";
     cout<<a<<" + "<<b<<" = "<<(a+b)<<endl;

     cout<<"Проверка операции -\n";
     cout<<a<<" - "<<b<<" = "<<(a-b)<<endl;

     cout<<"Проверка операции *\n";
     cout<<a<<" * "<<b<<" = "<<(a*b)<<endl;
     cout<<a<<" * "<<2<<" = "<<(a*2)<<endl;

     cout<<"Проверка операции /\n";
     cout<<a<<" / "<<2<<" = "<<(a/2)<<endl;

     cout<<"Проверка операции +=\n";
     a+=b;
     cout<<"a="<<a<<endl;

     cout<<"Проверка операции -=\n";
     a-=b;
     cout<<"a="<<a<<endl;

     cout<<"Проверка операции ==\n";
     cout<<a<<" == "<<b<<" = "<<(a==b)<<endl;

     cout<<"Проверка операции !=\n";
     cout<<a<<" != "<<b<<" = "<<(a!=b)<<endl;
```

```
    cout<<"Проверка операции !()\n";
    cout<<"!"<<a<<' '<<(!(a))<<endl;
    return 0;
}
```

Полученные результаты

Вывод
a=(4,3) b=(1,2)
Ввод a
(5,2)
a=(5,2)
Проверка операции +
(5,2) + (1,2) = (6,4)
Проверка операции -
(5,2) - (1,2) = (4,0)
Проверка операции *
(5,2) * (1,2) = 9
(5,2) * 2 = (10,4)
Проверка операции /
(5,2) / 2 = (2.5,1)
Проверка операции +=
a=(6,4)
Проверка операции -=
a=(5,2)
Проверка операции ==
(5,2) == (1,2) = 0
Проверка операции !=
(5,2) != (1,2) = 1
Проверка операции !()
!(5,2) 0

4 Листинг реализации класса

```
Vector &Vector::operator+=(const Vector&v) {
    x += v.x;
    y += v.y;
    return *this;
}

Vector operator+(Vector v1, Vector v2) {
    return v1 += v2;
}
Vector &Vector::operator-=(const Vector&v) {
    x -= v.x;
    y -= v.y;
    return *this;
}
```

```
Vector operator-(Vector v1, Vector v2) {
   return v1 -= v2;
}

double Vector::operator*(Vector v)
{
   return x*v.x+y*v.y;
}

Vector operator*(Vector v, double k)
{
   v.x *= k;
   v.y *= k;
   return v;
}

Vector operator/(Vector v, double k)
{
   v.x /= k;
   v.y /= k;
   return v;
}

bool operator==(const Vector &v1, const Vector &v2) {
   return (v1.x == v2.x and v1.y == v2.y);
}

bool operator!=(const Vector &v1, const Vector &v2) {
   return !(v1==v2);
}


istream &operator>>(istream &s, Vector &v) {
   char c[3];
   return s>>c[0]>>v.x>>c[1]>>v.y>>c[2];
}

ostream &operator<<(ostream &s, const Vector &v) {
   return s<<'('<<v.x <<","<<v.y<<')';
}
```