

```

//___файл my.h
#include<stdlib.h>

using namespace std;

class my
{
    int *x,n; //вектор и его длина

    public:

        my(){n=0;x=0;} //конструктор без параметров

        my(int ); //конструктор с параметром

        my (my &&v); //конструктор перемещения

        my (const my&v); //конструктор копии

        ~my(); //деструктор

        my operator +(const my &a); //сложение по координатам, длина = min

        my operator *(int); //умножение вектора на число справа, v*(int)

        my &operator =(const my &a); //присваивание копированием

        my &operator =(my &a); //присваивание перемещением

        my &operator --(void); //декремент --v

        my operator --(int); // декремент v--

        friend my operator * (int,const my &z); //умножение вектора на число слева (int)*v

        friend void pr(const my &); //печать
};

//___файл my.cpp
#include"my.h"

my::my(int N)
{
    n=N;

    x= new int[n];

    for (int i=0;i<N;i++)

        x[i]=rand()%5;

}

my::my (const my&v)//вызывается при создании нового объекта, память под который еще не
выделена
{

```

```

        n=v.n;

        x= new int[n];

        for(int i=0;i<n;i++)

            x[i]=v.x[i];
    }

    my::~~my()
    {

        if(x) delete []x;
    }

    my & my::operator=(const my &v)
    {

        if(this==&v)return *this; //самоприсваивание

        if(n<v.n) //если выделенной памяти может не хватить, выделяем новую
        {

            delete []x; x= new int [v.n];

        }

        n=v.n;

        for(int i=0;i<n;i++)

            x[i]=v.x[i];

        return *this;
    }

    my & my::operator=(my &&v)
    {

        if(this==&v)return *this;

        if (x)delete []x; // очистка ранее выделенной памяти

        x= v.x; //передаем ресурс другому

        n=v.n;

        v.x=0; //чтобы деструктор не очистил переданную память

        return *this;
    }

    my::my(my &&v)
    {

```

```

        n=v.n;

        x=v.x; //передаем ресурс другому

        v.x=0; //чтобы деструктор не очистил переданную память

    }

    my my::operator +(const my &a)
    {
        int y=n<a.n?n:a.n; my tmp(y); //строим временный объект

        for (int i=0;i<tmp.n;i++)

            tmp.x[i]=x[i]+a.x[i];

        return tmp; //ресурс временного объекта может быть передан другому
    }

    my operator * (int r,const my &z)
    {

        my tmp(z);

        for (int i=0;i<z.n;i++)

            tmp.x[i]*=r;

        return tmp;

    }

    my my::operator * (int r)
    {

        my tmp(*this);

        for (int i=0;i<n;i++)

            tmp.x[i]*=r;

        return tmp;

    }

    my & my::operator --()
    {

        n--;if(n<0)n=0;return *this;

    }

    my my::operator --(int )
    {

```

```

        my tmp(*this);

        n--;if(n<0)n=0;

        return tmp;
    }

void pr(const my &v)
{
    for (int i=0;i<v.n;i++)

        cout<<v.x[i]<<" ";

        cout<<"\n";
//____файл main.cpp
}

#include"my.h"

int main()
{
    my v1(15);

    pr(v1);

    {

        my v2=v1,v3; //конструктор копии
        v3=my(3)+my(2)+(-1)*my(2); //присваивание перемещением
        pr(v3);
        v3=v1+(-1)*v2+---v1*(-1)+v2; //копирование и присваивание перемещением
        pr(v3);
        pr(v2);

    }

    v1=--v1; //присваивание копированием
    pr(v1+v1);

    return 0;

return 0;

}

```