# Lab 5. Przeciążenie operatorów binarnych

Klasa `my_vector` przedstawia wektor w 3D.

```
#include <iostream>

class my_vector
{
private:
    double* cr;  //cr[0] - x; cr[1] - y; cr[2] - z
public:
    my_vector() : cr(NULL) {}
    my_vector(double xx, double yy, double zz);
    ~my_vector() {
        if (cr)
            delete[] cr;
        cr = NULL;
    }

    my_vector(const my_vector& ob);
    my_vector(my_vector&& ob);
    my_vector& operator = (const my_vector& pb);
    my_vector& operator = (my_vector&& pb);

    my_vector operator + (const my_vector& right) const;
    my_vector operator * (const my_vector& right) const;   //vector product: res = v1 x v2
    double operator * (const my_vector* right) const;      //dot product: dot = v1 * v2

    friend std::istream& operator >> (std::istream& strm, my_vector& v);
    friend std::ostream& operator << (std::ostream& strm, const my_vector& v);
```

```
    void disp(const char* str);
private:
    void alloc();  //allocation of memory
};
```

Dla podanej klasy napisz implementacje wszystkich operatorów oraz metod w taki sposób, żeby podana poniżej funkcja *main* działała poprawnie. Metoda *alloc* służy dla dynamicznego alokowania pamięci.

```
void  my_vector::alloc()
{
    try
    {
        cr = new double[3];
        memset(cr, 0, 3 * sizeof(double));
    }
    catch (std::bad_alloc)
    {
        cout << "memory allocation error" << endl;
        system("pause");
        exit(1);
    }
}
```

Metoda *disp*(…) wymaga przeciążenia operatora << wyprowadzającego do strumienia *cout*.

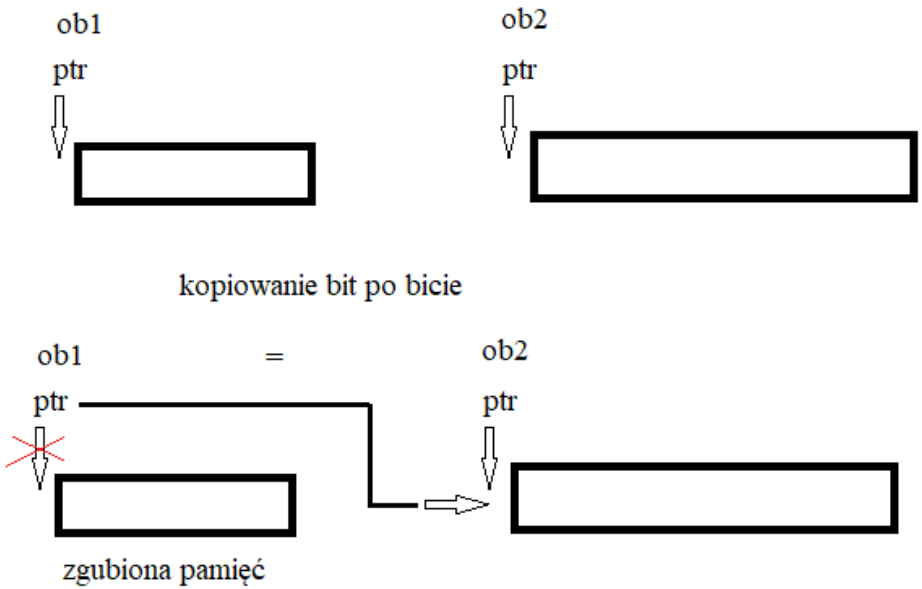
```
void my_vector::disp(const char *str)
{
    if (cr)
    {
         cout << str << ":" << *this << endl;
    }
}


int main()
```

```
{
    my_vector v1(1, 2, -1), v2(2, 1, 1), res;

    v1.disp("v1");
    v2.disp("v2");

    res = v1 + v2;
    res.disp("v1 + v2");

    my_vector res1 = v1 * v2;
    res1.disp("v1 x v2");

    res1 = v2 * v1;
    res1.disp("v2 x v1");

    cout << "v1 * v2: " << v1 * (&v2) << endl;

    my_vector v3(0, 0, 0);
    cin >> v3;
    v3.disp("v3");

    ofstream my_file("my_output_file.txt");
    if (my_file.is_open())
    {
        my_file << "v1 * v2: " << v1 * (&v2);
        my_file.close();
        system("notepad.exe my_output_file.txt");
    }

    return 0;
}
```

Przeciążenie operatora = :

Fragment kodu:

## Brak przeciążenia operatora = :




1. Zgubiona pamięć
2. Ob1 i ob2 współdzielą ten samy bufor w pamięci heap.
3. Wywołanie destruktora spowoduje błąd przy zwolnieniu pamięci alokowanej dynamicznie.

**Poprawne rozwiązanie – przeciążamy operator = :**