```
// W4_1.cpp : Defines the entry point for the console application.
//            Operator przypisania.

#include "stdafx.h"
#include <iostream>
using namespace std;

class aa_class
{
      double *arr;       //tablica
      int dim;           //rozmiar tablicy
      char name[256];
public:
      aa_class(int dd, char *str, double start_val);  //konstruktor
      ~aa_class();
      void disp();
};

aa_class::aa_class(int dd, char *str, double start_val)
{
      dim = dd;
      try
      {
            arr = new double [dim];
      }
      catch(bad_alloc aa)
      {
            cout << "zamalo pamieci\n";
            system("pause");
            exit(1);
      }

      int i;
      for(i=0; i<dim; i++)
            arr[i] = start_val+i;

      strcpy_s(name, sizeof(name), str);
}

aa_class::~aa_class()
{
```

```
	delete [] arr;
}

void aa_class::disp()
{
	int i;
	cout << name << endl;
	cout << "adres arr: " << arr << endl;
	cout << "i\t\t" << "arr" << endl;
	for(i=0; i<dim; i++)
	{
		cout << i << "\t" << arr[i] << endl;
	}
}


int _tmain(int argc, _TCHAR* argv[])
{
	aa_class ob1(10, "obiekt 1", 1.0), ob2(10, "obiekt 2", 10.0);

	ob1.disp();
	ob2.disp();

	//UWAGA! Po wykonani tego operatora wartosc wskaznika do tablicy ob1.arr bedzie
	//przepisana wartoscia wskaznika do tablicy ob2.arr
	// - pamiec ob1.arr pozostaje zgubiona i nigdy nie bedzie zwolniona
	// - przy wykonani destruktora obiektu ob1 (wskaznik ob1.arr wskazuje do ob2.arr)
	//   bedzie zwolniona pamiec ob2.arr; przy wykonani destruktora obiektu ob2
	//   (wskaznik ob2.arr wskazuje na ob2.arr) ta sama pamiec bedzie zwolniona
	//   podwujnie - page fault!
	ob1 = ob2;
//sprawdzic pod debugerem adresy arr, name
	ob1.disp();
	system("pause");
	return 0;
}
```