```cpp
// W6.cpp : Defines the entry point for the console application.
//          Konstruktor kopiujacy przy wywolani funkcji

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

class myclass;    //to jest referencje zapowiadajaca do klasy
void fun(myclass s);

class myclass
{
    double *ptr;
    int it;
public:
    myclass() {
        cout << "Wywolanie konstruktora\n";
        it = 0; ptr = NULL;
    }
    myclass(int items);
    myclass(const myclass &obj);
    ~myclass();
    void put(double a);
    int getnoitems();
    double get(int i);
};

myclass::myclass(int items)
/*=============================================================================
Konstruktor klasy
=============================================================================*/
{
    cout << "Wywolanie konstruktora sparametryzowanego\n";
    try
    {
        ptr = new double [items];
    }
    catch(...)
    {
        cout << "blad alokacji pamieci\n";
        system("pause");
        exit(1);
    }
    it = 0;
    memset((void *)ptr, 0, items*sizeof(double));
}

myclass::myclass(const myclass &obj)
/*=============================================================================
Konstruktor kopiujacy
=============================================================================*/
{
    cout << "Wywolanie konstruktora kopiujacego\n";

    it = obj.it;

    if(obj.ptr)
    {
        size_t noitems = _msize(obj.ptr);  //rozmiar tablicy
        int items = (int)noitems/sizeof(double);
        try
        {
            ptr = new double [items];
        }
        catch(...)
        {
            cout << "blad alokacji pamieci\n";
            system("pause");
            exit(1);
        }

        memcpy((void *)ptr, (const void *)obj.ptr, items*sizeof(double));
    }
```

```
    else
    {
        ptr = NULL;
    }
}

myclass::~myclass()
{
    cout << "wywolanie destruktora:  ";

    if(ptr)
    {
        delete [] ptr;
        ptr = NULL;
        cout << "  free ptr";
        it = 0;
    }

    cout << "\n";
    system("pause");
}

int myclass::getnoitems()
{
    if(!ptr)
        return -1;

    size_t noitems = _msize(ptr);  //rozmiar tablicy
    return (int)noitems/sizeof(double);
}

void myclass::put(double a)
{
    if(ptr)
    {
        int noitems = getnoitems();  //rozmiar tablicy
        if(it >= noitems)
        {
            cout << "tablica jest wypelniona\n";
            return;
        }

        ptr[it] = a;
        it++;
    }
}

double myclass::get(int i)
{
    bool IsOK = 1;

    if(ptr)
    {
        int noitems = getnoitems();
        if(i < 0 || i >= noitems)
            IsOK = 0;
    }
    else
        IsOK = 0;

    if(!IsOK)
    {
        cout << "indeks i przekroczyl rozmiar tablicy\n";
        system("pause");
        exit(1);
    }

    return ptr[i];
}


int _tmain(int argc, _TCHAR* argv[])
{
    myclass ob(10);
```

```
    ob.put(0.);
    ob.put(1.0);
    ob.put(2);

    fun(ob);

    myclass ob1;
    fun(ob1);

    return 0;
    //tu destruktor bedzie wywolany
}

void fun(myclass s)
{
    //kompilator tworzy kopie obiektu myclass s
    //wywoluje sie konstruktor kopiujacy, ktory tworzy tablice ptr w innych adresach
    pamieci.
    //To zapobiega bladu przy wywolani destruktora (przy wyjsciu z ciala funkcji)

    int noitems = s.getnoitems();
    int i;
    double aa;
    char str[32];

    for(i=0; i<noitems; i++)
    {
        aa = s.get(i);
        sprintf(str, "%lf", aa);

        cout << "i=  " << i << "\t" << str << "\n";
    }

    //Tu bedzie wywolany destruktor -> pamiec dla ptr (adresy kopii) pozostanie zwolniona
}
```