```cpp
// allocator.cpp : Defines the entry point for the console application.
//          my_allocator, my_deallocator zmieniaja wartosc wskaznikow
//      Dla tego przekazujemy funkcjom wskazniki do wskaznikow
//      W przypadku przekazywania wskaznikow w steku powstawala by kopia tych
//      wskaznikow, a przy wyjsciu z funkcji ta kopia byla by niszczona -
//      program main nie dostal by wartosci tych wskaznikow

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

class MY_CLASS;
void my_allocator(double **ptr, MY_CLASS **pCl, size_t noitems);
void my_deallocator(double **ptr, MY_CLASS **pCl);

class MY_CLASS
{
      int *i_ptr;
public:
      MY_CLASS(size_t dim);
      ~MY_CLASS();
};

MY_CLASS::MY_CLASS(size_t dim)
{
      try
      {
            i_ptr = new int [dim];
            for(size_t it=0; it<dim; ++it)
                  i_ptr[it] = (int)(it);
      }
      catch(bad_alloc xx)
      {
            cerr << "mem alloc err\n";
            system("pause");
            exit(1);
      }
}

MY_CLASS::~MY_CLASS()
{
      if(i_ptr)
            delete [] i_ptr;
```

```
}

int _tmain(int argc, _TCHAR* argv[])
{
	double *pptr = NULL;
	MY_CLASS *pMyCl = NULL;
	size_t ddim = 4;
	my_allocator(&pptr, &pMyCl, ddim);
	for(size_t i=0; i<ddim; ++i)
		pptr[i] = (double)(i);
	my_deallocator(&pptr, &pMyCl);
	system("pause");
	return 0;
}

void my_allocator(double **ptr, MY_CLASS **pCl, size_t noitems)
{
	try
	{
		*ptr = new double [noitems];
		*pCl = new MY_CLASS (noitems);
	}
	catch(bad_alloc xx)
	{
		cout << "mem alloc error\n";
		system("pause");
		*ptr = NULL;
	}
}

void my_deallocator(double **ptr, MY_CLASS **pCl)
{
	if(*ptr)
		delete [] *ptr;
	*ptr = NULL;
	if(*pCl)
		delete *pCl;
	*pCl = NULL;
}
```