```
// W16.cpp : Defines the entry point for the console application.
//		Praca z plikiem I/O: jeden plik - dwa strumieni
//					Wprowadzenie w exception handlers

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

namespace MY_ENUM
{
	enum MY_ERROR
	{
		MY_EXCPT_ALLOC_MEM,
		MY_EXCEPT_OPEN_FILE,
		MY_EXCEPT_ACCESS_FILE,
		MY_EXCEPT_TOT_NUMB
	}my_err;

	enum MY_WARNING
	{
		MY_WARN_NO_EQ,
		MY_WARN_TOT_NUMB
	}my_warn;
};

using namespace MY_ENUM;

class CExcept
{
	int pos;  //wskazuje do numera wierszy tekstowego w tabeli
			//tab_err lub tab_warn
	int typ;  //0 - error (tabela tab_err), 1 - warning (tabela tab_warn)
	static char *tab_err[MY_EXCEPT_TOT_NUMB];
	static char *tab_warn[MY_WARN_TOT_NUMB];
```

```
public:
	CExcept::CExcept(enum MY_ERROR my_err);
	CExcept::CExcept(enum MY_WARNING my_warn);
	int get_typ(){return typ;}

	void except_err();
	void except_warn();
};

CExcept::CExcept(enum MY_ERROR my_err)
{
	pos = my_err;
	typ = 0;
}

CExcept::CExcept(enum MY_WARNING my_warn)
{
	pos = my_warn;
	typ = 1;
}

void CExcept::except_err()
{
	cout << tab_err[pos] << endl;
	system("pause");
	exit(1);
}

void CExcept::except_warn()
{
	cout << tab_warn[pos] << endl;
	system("pause");
}

//Inicjowanie statycznych skladowych klasy jako zmiennych globalnych
char *CExcept::tab_err[] =
```

```
{
	"Memory allocation error",
	"Open file error",
	"Access file eror"
};

char *CExcept::tab_warn[] =
{
	"Objects are not equal",
};

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////

class coord
{
	double x, y;
public:
	coord(double xx, double yy) : x(xx), y(yy) {};
	coord() { x = y = 0;}
	void set(double xx, double yy) {x = xx; y = yy;}
	friend ofstream & operator << (ofstream &strm, const coord *cr);
	friend ifstream & operator >> (ifstream &strm, coord *cr);
	bool operator != (const coord &ob2)
	{
		return (fabs(x-ob2.x) > 1.0e-10 || fabs(y-ob2.y) > 1.0e-10);
	}
};

ofstream & operator << (ofstream &strm, const coord *cr)
{
	strm.write(reinterpret_cast<const char *>(cr), static_cast<streamsize>(sizeof(coord)));
	if(strm.bad() || strm.fail())
	{
		CExcept exp(MY_EXCEPT_ACCESS_FILE);
		throw exp;
```

```
	}
	return strm;
}
ifstream & operator >> (ifstream &strm, coord *cr)
{
	strm.read(reinterpret_cast<char *>(cr), static_cast<streamsize>(sizeof(coord)));
	if(strm.bad() || strm.fail() || strm.gcount() != sizeof(coord))
	{
		CExcept exp(MY_EXCEPT_ACCESS_FILE);
		throw exp;
	}
	return strm;
}

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
int _tmain(int argc, _TCHAR* argv[])
{
	int i, imax = 1000000;
	char filename[] = "filedata.bin";
	coord *pcoord;
	coord *pcoord1;
	ofstream fl_out;
	ifstream fl_in;

	try{

		//test of exceptions
		//CExcept exc1(MY_EXCEPT_OPEN_FILE);
		//throw exc1;
		//CExcept exc2(MY_EXCEPT_ACCESS_FILE);
		//throw exc2;

		//CExcept exc3(MY_WARN_NO_EQ);
		//exc3.except_warn();
```

```
fl_out.open(filename, ios_base::out | ios_base::binary);
fl_in.open(filename, ios_base::in | ios_base::binary);

if(!fl_out.is_open() || !fl_in.is_open())
{
        CExcept exc1(MY_EXCEPT_OPEN_FILE);
        throw exc1;
}

try{
        pcoord = new coord [imax];
        pcoord1 = new coord [imax];
        //for test of exception handlers
        //bad_alloc xx;
        //throw xx;
}
catch(bad_alloc aa)
{
        CExcept exc(MY_EXCPT_ALLOC_MEM);
        throw exc;
}

for(i=0; i<imax; i++)
        pcoord[i].set(static_cast<double>(i), 10.0);

for(i=0; i<imax; i++)
{
        fl_out << &pcoord[i];
        streamsize posp = fl_out.tellp();
}

//uwaga! to jest konieczne, dla tego ze czesc danych pozostaje w
//buforze systemowym, a nie w pliku
//To jest zle dla tego, ze w zlozonych algorytmach zapisu-odczytu
//trzeba czesto flaszowac bufor strumienia fl_out!
```

```
fl_out.flush();

for(i=0; i<imax; i++)
{
	streamsize posg = fl_in.tellg();
	fl_in >> &pcoord1[i];
	//for test of operator !=
	//pcoord[i].set(0, 123);
}

for(i=0; i<imax; i++)
{
	if(pcoord[i] != pcoord1[i])
	{
		cout << "error!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n";
		break;
	}
}

delete [] pcoord;
delete [] pcoord1;

fl_out.close();
fl_in.close();

}//end try block
catch(CExcept exc)
{
	if(exc.get_typ() == 0)
		exc.except_err();
	else
		exc.except_warn();
}

system("pause");
```

```
        return 0;
}
```