Lab_7

Utwórz nowy projekt, który zawiera:

- Klasę-szablon my_vect (file my_vect.h)
- Klasę my_mess dla obsługi błędów I komunikatów (my_mess.h, my_mess.cpp)
- Klasę interfejsu, która łączy pomiędzy sobą szablon i dane oraz udostępnia użytkownikowi interfejs na poziomie aplikacji konsolowej (my_interf.h, my_interf.cpp).
- Klasę danych my_coord (my_coord.h, my_coord.cpp)

Klasa my_vect:

```
        T *dat;         //pointer to the data of generic type, array of container
        size_t ndim;    //dimension of array to be allocated dynamically
        size_t last;    //index of the first empty position in array
        my_mess msg;    //declaration of the message object

public:
        my_vect(size_t dm); //constructor - allocates memory for array of T type
                            //on dm elements.
        my_vect() { dat = NULL; ndim = last = 0; }
        ~my_vect();         //release of array
        void init(size_t dm);       //allocates memory for array with dm elements,
                                    //if dat = NULL (after default constructor)
        T *get_begin()              //returns pointer T * to the begin of array
        T *get_end();               //returns pointer T * to the dat[last]
        void disp();                //displays an array to monitor
        void push(const T &ob);     //add an element of the type T to first empty position
                                    //of array
        T *pop();                   //returns pointer to the last element of the filled
                                    //part of array or NULL, if an array is empty.
        T & operator [] (const size_t ind); //0 <= ind < last
        void remove(size_t ind);    //erases an element of array with index ind and
                                    //compresses array
private:
        bool alloc();               //dynamically allocates of array
        bool realloc();             //enlarge the dimension ndim of array and reallocate
                                    //memory.
```

Dodaj realizację klasy szablonu i umieść funkcję szablonu Find (...) w tym pliku.
Napisz obsługę błędów i komunikatów w plikach my_mess.cpp i my_mess.h. Możesz skorzystać się z przykładu do wykładów W29 albo wprowadzić swój własny system obsługi.

my_mess.h:

```
class my_mess
{
public:
        enum MY_MESSAGE
        {
                ERR_ALLOC_MEM,      //memory allocation error
                WARN_ARR_FULL,      //array is full (last > ndim)
                WARN_ARR_EMPT,      //array is empty (last = 0)
                WARN_ARR_UNKN,      //invalid operation (interface)
                TOT_NUMB            //total number of messages
        };

        static char *strtab[TOT_NUMB];  //array of messages
```

```
public:
	void mess(enum MY_MESSAGE ms);	// When this function is called, the message
					//appears on the screen. If this is an error,
					//the calculation must be finished, if not,
					//continue. ms - enumeration member that gives
					//the message number};
```

my_mess.cpp:

```
char *my_mess::strtab[] =
{
	"E  memory alloc error",		//ERR_ALLOC_MEM
	"W  array is full",			//WARN_ARR_FULL
	"W  array is empty",			//WARN_ARR_EMPT
	"W  invalid operation"			//WARN_ARR_UNKN
};

void my_mess::mess(enum MY_MESSAGE ms)
{
	if(strtab[ms][0] == 'E')
	{
		cout << "ERROR: " << &strtab[ms][1] << endl;
		system("pause");
		exit(1);
	}
	else if(strtab[ms][0] == 'W')
	{
		cout << "WARNING: " << &strtab[ms][1] << endl;
	}
}
```

Przykładowy wariant klasy interfejsu. Konieczne jest użycie przeciążenia operatorów <<, >> przy wprowadzeniu/wyprowadzeniu obiektów danych do/z strumieni cin, cout.

my_interf.h:

```
#include "my_vect.h"
#include "my_coord.h"

class my_interf
{
	enum MY_INTERF
	{
		MY_INTERF_PUSH,
		MY_INTERF_POP,
		MY_INTERF_DISP,
		MY_INTERF_FIND,
		MY_INTERF_FINISH,
		MY_INTERF_TOT
	};

	my_vect<my_coord> vect;
	static char *str_interf[];  // messages displayed on the monitor
	my_mess msg;
public:
	bool run;

	my_interf();
	my_interf(size_t dim);
	void menu();
	void push();
	void pop();
```

```
        void disp();
        void find();
        void finish();
        void defaul();
};
```

my_interf.cpp:

```
my_interf::my_interf(size_t dim)
{
        vect.init(dim);
        run = true;
}

void my_interf::menu()
{
        int i;
        for(i=0; i<MY_INTERF_TOT; i++)
        {
                cout << str_interf[i] << endl;
        }
}

void my_interf::push()
{
        my_coord ob(0,0);
        cin >> ob;
        vect.push(ob);
}

void my_interf::pop()
{
        my_coord *ptr = NULL;
        ptr = vect.pop();
        if(ptr)
        {
                cout << *ptr;
        }
        else
        {
                msg.mess(my_mess::WARN_ARR_EMPT);
        }
}

void my_interf::disp()
{
        vect.disp();
}

void my_interf::find()
{
        my_coord ob(0, 0);
        my_coord *ptr = NULL;
        cout << "input x, y - object for search\n";
        cin >> ob;
        ptr = vect.get_begin();
        size_t dist;
        while(ptr)
        {
                ptr = Find(ptr, vect.get_end(), ob);
                if(ptr)
                {
                        dist = ptr-vect.get_begin();
                        cout << "it = " << dist << " " << *ptr;
                        ptr++;
```

```
			}
			else
				cout << "search end\n";
		}
}
```

Klasa danych my_coord. Tą klasę trzeba uzupełnić.

```
class my_coord
{
	double *pcoord;  //pcoord[0] - x coordinate; pcoord[1] - y coordinate
	my_mess msg;
public:
	my_coord() { alloc(); pcoord[0] = pcoord[1] = 0; }
	my_coord(double x, double y);
	~my_coord();
private:
	void alloc();
};
```

Funkcja main:

```
int _tmain(int argc, _TCHAR* argv[])
{
	int op;  //operacje
	my_interf ifc(1000);

	while(ifc.run)
	{
		ifc.menu();
		cin >> op;
		switch(op)
		{
		case MY_INTERF_PUSH: //push
			ifc.push();
			break;
		case MY_INTERF_POP:
			ifc.pop();
			break;
		case MY_INTERF_DISP:
			ifc.disp();
			break;
		case MY_INTERF_FIND:
			ifc.find();
			break;
		case MY_INTERF_FINISH:
			ifc.finish();
			break;
		default:
			ifc.defaul();
		};
	}

	system("pause");
	return 0;
}
```

Jako wzorzec, można użyć przykłady do wykładów W23, W29 (wyjątki, praca z plikiem binarnym).