Автор: Єлєсін Артем , КІТ-119б

Дата: 01.06.2020

Лабораторна робота 10. ШАБЛОННІ ФУНКЦІЇ

Тема. Шаблонні функції.

Мета – отримати базові знання про шаблонізацію (узагальнення) на основі шаблонних функцій.

Загальне завдання

Створити клас, який не має полів, а всі необхідні дані передаються безпосередньо у функції. Клас має виконувати такі дії:

- виводити вміст масиву на екран;

- визначати індекс переданого елемента в заданому масиві;

- сортувати елементи масиву;

- визначати значення мінімального елемента масиву. При цьому необхідно продемонструвати роботу програми як з використанням стандартних типів даних, так і типів, створених користувачем.

Опис класів

Клас с основним завданням: CList

Опис методів

template <class T1, class T2> void showMass(T1, T2); - показати данні

template<class T1, class T2, class T3> T2 index(T1, T3, T2); - індекс елементу

template<class T1, class T2, class T3>void sortMass(T1, T2, T3); - сортувати масив

template<class T1, class T2, class T3> T3 minEl(T1 mass, T2 size, T3 Rrimer); - мінімальний елемент

Текст програми

CList.cpp

```cpp
#include "CList.h"

template<class T1, class T2>
void CList::showMass(T1 mass, T2 size)
{
	for (T2 i = 0; i < size; i++) {
		cout<<mass[(int)i]<<endl;
	}
}

template<class T1, class T2, class T3>
T2 CList::index(T1 mass,T3 el,T2 size)
{
	for (T2 i = 0; i < size; i++) {
		if (el == mass[(int)i])
			return i;
	}
	return -1;
}

template<class T1, class T2, class T3>
void CList::sortMass(T1 mass, T2 size, T3 Primer)
{
	T3 temp;
	bool prz=1;
	while (prz) {
		prz = 0;
		for (T2 i = 0; i < size - 1; i++) {
			if (mass[(int)i] > mass[(int)i + 1])
			{
				temp = mass[(int)i];
				mass[(int)i] = mass[(int)i + 1];
				mass[(int)i + 1] = temp;
				prz = 1;
			}
		}
	}
}

template<class T1, class T2, class T3>
T3 CList::minEl(T1 mass, T2 size, T3 Primer)
{
	T3 min = Primer;
	if (size > 0)
	min = mass[0];
	for (T2 i = 0; i < size; i++) {
		if (min > mass[(int)i])
		{
			min = mass[(int)i];
		}
	}
	return min;
}
```

CList.h

```cpp
#pragma once
#include <iostream>
```

```
using std::cout;
using std::cin;
using std::endl;
using std::ostream;
using std::istream;
using std::string;
class CList
{
public:
	template <class T1, class T2> void showMass(T1, T2);
	template<class T1, class T2, class T3> T2 index(T1, T3, T2);
	template<class T1, class T2, class T3>   void sortMass(T1, T2, T3);
	template<class T1, class T2, class T3> T3 minEl(T1 mass, T2 size, T3 Rrimer);
};
```

Sourse.cpp

```
#include "CList.cpp"
using std::ostream;
struct MYTYPE {
	string ch;

	//MYTYPE& operator=(MYTYPE& temp) {
		//ch = temp.ch;
		//return *this;
	//};
};
ostream& operator<<(ostream& output, MYTYPE& obj)
{
	output << obj.ch;
	return output;
};
istream& operator>>(istream& input, MYTYPE& obj)
{
	input >> obj.ch;
	return input;
};
bool operator==(MYTYPE& a,MYTYPE& b) {
	return a.ch == b.ch;

};
bool operator>(MYTYPE& a, MYTYPE& b) {
	return a.ch > b.ch;

};
int main() {
	{
	CList list;
	int d[8] = { 10,9,7,5,6,844,0,1};
	MYTYPE b;
	b.ch = "CCCC";
	MYTYPE a[4] = { "BBBB", "DDDD", "CCCC" , "AAAA" };
	list.showMass(a, 4);
	cout << list.index(a, b, 4) << endl;
```

```
	list.sortMass(a, 4, b);
	list.showMass(a, 4);
	b = list.minEl(a, 4, a[0]);
	cout << b << endl;
	list.showMass(d, 8);
	list.sortMass(d, 8, d[0]);
	list.showMass(d, 8);
}
	if (_CrtDumpMemoryLeaks())
		cout << "\nMemory leack deteckted\n";
	else
		cout << "\nMemory is not leack deteckted\n";

}
```

Test.cpp

```
#include "CList.cpp"
int main() {
	{
		CList list;
		int test[8] = { 9,7,5,6,844,0,1,0 };
		int rez1[8] = { 0,0,1,5,6,7,9,844 };
		int rez2 = 0;
		list.sortMass(test, 8, test[0]);
		for (int i = 0; i < 8; i++) {
			if (test[i] == rez1[i])
				cout << "test 1." << i << ": true"<<endl;
			else
				cout << "test 1." << i << ": false" << endl;
		}
		if(list.minEl(test,8,test[0])==rez2)
			cout << "test 2: true" << endl;
		else
			cout << "test 2: false" << endl;
		if(list.index(test,test[5],8)==5)
			cout << "test 3: true" << endl;
		else
			cout << "test 3: false" << endl;
	}
	if (_CrtDumpMemoryLeaks())
		cout << "\nMemory leack deteckted\n";
	else
		cout << "\nMemory is not leack deteckted\n";

}
```

Висновок

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з шаблонними функціями.

Було розроблено програму, що працює з шаблонними функціями.

Шаблонні функції та взагалі шаблони призначені для роботи з змінними невідомого заздалегідь типу. Це допомагає скоротити код завдяки відсутності перевантажень функцій.

Програма протестована, витоків пам’яті немає, виконується без помилок.