Автор: Татаренко А., КІТ-119а

Дата: 17 червня 2020

Лабораторна робота №15.

РОЗУМНІ ВКАЗІВНИКИ

Тема. Розумні вказівники.

Мета – по результатах практичної роботи порівняти розумні вказівники бібліотеки STL.

1 Завдання до роботи

Індивідуальне завдання 19.

Створити STL-контейнер, що містить у собі об’єкти ієрархії класів, використати розумні вказівники:
- auto_ptr;
- unique_ptr;
- shared_ptr;
- weak_ptr.

2 Розробка алгоритму розв’язання задачі.

2.1 Опис змінних

Arr stud_array; class Student; class Arr;

Класи, методи, функції, конструктори

3 Код програми

class1.h

```
#pragma once
#include "Header.h"

class Student
{
protected:
	int age;
	int number_stud;
	int middle_mark;
	string name;
	bool debt;
	int prog_d;

public:
	virtual string get_info() const;
	virtual stringstream get_str() const;
	int get_numb() const;
	virtual bool elementOutput(int, string);
	virtual int countElement(int, string);

	Student();
	Student(int, int, int, string, bool, int);
	Student(const Student&);
	virtual ~Student();

	friend ostream& operator<< (ostream&, const Student&);
	virtual bool operator==(const int) const;

};
```

class2.h

```
#pragma once
#include "class1.h"

class Course final : public Student
{
private:
	int course;

public:
	string get_info() const override final;
	stringstream get_str() const override final;
	bool elementOutput(int, string) override final;
	int countElement(int, string) override final;

	Course();
	Course(int, int, int, string, bool, int, int);
	Course(const Course&);
	~Course() override final;

	bool operator==(const int) const override final;

};
```

Functor.h

```
#pragma once

#include "class1.h"
```

```
class Functor
{
private:
	int value;

public:
	bool operator()(const shared_ptr<Student>&, const shared_ptr<Student>&);

	Functor(int);
	~Functor();
};
```

Header.h

```
#pragma once

#define _CRT_SECURE_NO_WARNINGS
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h>
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE)

#include <string>
#include <iostream>
#include <iomanip>
#include <locale>
#include <fstream>
#include <sstream>
#include <istream>
#include <vector>
#include <memory>
#include <list>
#include <map>
#include <set>
#include <unordered_set>
#include <algorithm>
#include <iterator>

using std::string;
using std::cin;
using std::cout;
using std::endl;
using std::setw;
using std::boolalpha;
using std::setiosflags;
using std::ios;
using std::ifstream;
using std::ostream;
using std::ofstream;
using std::stringstream;
using std::istream;
using std::vector;
using std::list;
using std::map;
using std::set;
using std::unordered_set;
using std::unique_ptr;
using std::shared_ptr;
using std::auto_ptr;
using std::weak_ptr;
using std::advance;
using std::stoi;
using std::for_each;
using std::make_move_iterator;
```

```
using std::set_intersection;
using std::back_inserter;
using std::pair;
using std::transform;
using std::inserter;
using std::make_shared;
using std::make_unique;
```

PointerCount.h

```
#pragma once

class PointerCount
{
private:
	int pointerCount = 0;

public:
	void Increment() { ++pointerCount; }

	int Decrement() { return --pointerCount; }

	int GetPointerCount() const { return pointerCount; }
};
```

SmartPointer.h

```
#pragma once

#include "Header.h"
#include "PointerCount.h"

template <typename T>
class SmartPointer
{
private:
	T* object{ nullptr };
	PointerCount* pointerCount{ nullptr };

public:

	SmartPointer<T>& operator=(const SmartPointer<T>& pointer)
	{
		if (this != &pointer)
		{
			if (pointerCount && pointerCount->Decrement() == 0)
			{
				delete pointerCount;
				delete object;
			}

			object = pointer.object;
			pointerCount = pointer.pointerCount;
			pointerCount->Increment();
		}
		cout << "Присваивание умного указателя. Кол-во ссылок: " << pointerCount-
>GetPointerCount() << endl;

		return *this;
	}
	T& operator*()
	{
```

```
                return *object;
        }
        T* operator->()
        {
                return object;
        }

        SmartPointer(T* object) : object{ object }, pointerCount{ new PointerCount() }
        {
                pointerCount->Increment();
                cout << "Создан умный указатель. Кол-во ссылок : " << pointerCount-
>GetPointerCount() << endl;
        }
        virtual ~SmartPointer()
        {
                if (pointerCount)
                {
                        int decrementCount = pointerCount->Decrement();
                        cout << "Умный указатель уничтожен. Кол-во ссылок: " <<
decrementCount << endl;
                        if (decrementCount <= 0)
                        {
                                delete pointerCount;
                                delete object;
                                pointerCount = nullptr;
                                object = nullptr;
                        }

                }
        }
        SmartPointer(const SmartPointer<T>& other) :object{ other.object }, pointerCount{
other.pointerCount }
        {
                pointerCount->Increment();
                cout << "Умный указатель скопирован. Кол-во ссылок: " << pointerCount-
>GetPointerCount();
        }
};
```

class1.cpp

```
#include "class1.h"

string Student::get_info() const
{
        stringstream temp;

        temp.setf(std::ios::left);
        temp << setw(10) << age << setw(8) << number_stud << setw(16) << middle_mark <<
setw(9)
                << name << setw(7) << debt << setw(14) << prog_d;

        return temp.str();
}

int Student::get_numb() const
{
        return number_stud;
}

stringstream Student::get_str() const
{
        stringstream temp;
        temp << " " << age << " " << number_stud << " " << middle_mark << " "
```

```
		<< name << " " << debt << " " << prog_d;

	return temp;
}

int Student::countElement(int value, string data)
{
	try
	{
		if (value == 1)
		{
			if (this->name == data)
				return 1;
			else
				return 0;
		}
		else if (value == 2)
		{
			int number = stoi(data);
			if (this->age == number)
				return 1;
			else
				return 0;
		}
		else if (value == 3)
		{
			int number = stoi(data);
			if (this->middle_mark == number)
				return 1;
			else
				return 0;
		}
		else if (value == 4)
		{
			int number = stoi(data);
			if (this->prog_d == number)
				return 1;
			else
				return 0;
		}
		else if (value == 5)
		{
			int number = stoi(data);
			if (this->number_stud == number)
				return 1;
			else
				return 0;
		}
		else if (value == 6)
		{
			int number = 0;
			if (data == "true" || data == "true" || data == "1")
				number = 1;
			else
				number = 0;

			if (this->debt == number)
				return 1;
			else
				return 0;
		}
	}
	catch (const std::exception& ex)
	{
		cout << ex.what() << endl;
```

```
			return 0;
		}

		return 0;
}

bool Student::elementOutput(int value, string data)
{
		try
		{
			if (value == 1)
			{
				if (this->name == data)
					cout << *this << endl;
				return true;
			}
			else if (value == 2)
			{
				int number = stoi(data);
				if (this->age == number)
					cout << *this << endl;
				return true;
			}
			else if (value == 3)
			{
				int number = stoi(data);
				if (this->middle_mark == number)
					cout << *this << endl;
				return true;
			}
			else if (value == 4)
			{
				int number = stoi(data);
				if (this->prog_d == number)
					cout << *this << endl;
				return true;
			}
			else if (value == 5)
			{
				int number = stoi(data);
				if (this->number_stud == number)
					cout << *this << endl;
				return true;
			}
			else if (value == 6)
			{
				int number = 0;
				if (data == "true" || data == "true" || data == "1")
					number = 1;
				else
					number = 0;

				if (this->debt == number)
					return 1;
				else
					return 0;
			}
		}
		catch (const std::exception& ex)
		{
			cout << ex.what() << endl;
			return 0;
		}

		return 0;
```

```
}

ostream& operator<< (ostream& output, const Student& other)
{
	output << other.get_info();
	return output;
}

bool Student::operator==(const int ns) const
{
	return this->number_stud == ns;
}

Student::Student(int a, int n, int m, string na, bool d, int pd) : age(a),
number_stud(n), middle_mark(m), name(na), debt(d), prog_d(pd)
{
	//cout << "\nВызвался конструктор с параметрами";
}
Student::Student() : age(17), number_stud(0), middle_mark(8), name("Bond"), debt(1),
prog_d(15)
{
	//cout << "\nВызвался конструктор по умолчанию.";
}
Student::Student(const Student& other) : age(other.age), number_stud(other.number_stud),
middle_mark(other.middle_mark), name(other.name), debt(other.debt), prog_d(other.prog_d)
{
	//cout << "\nВызвался конструктор копирования.";
}
Student::~Student()
{
	//cout << "\nВызвался деструктор";

}
```

class2.cpp

```
#include "class2.h"

stringstream Course::get_str() const
{
	stringstream temp;

	temp << " " << age << " " << number_stud << " " << middle_mark << " "
		<< name << " " << debt << " " << prog_d << " " << course;

	return temp;
}

string Course::get_info() const
{
	stringstream temp;

	temp.setf(ios::left);
	temp << setw(10) << age << setw(8) << number_stud << setw(16) << middle_mark <<
setw(9)
		<< name << setw(7) << debt << setw(14) << prog_d << setw(4) << course;

	return temp.str();
}

int Course::countElement(int value, string data)
{
	try
	{
		if (value == 1)
```

```
	{
		if (this->name == data)
			return 1;
		else
			return 0;
	}
	else if (value == 2)
	{
		int number = stoi(data);
		if (this->age == number)
			return 1;
		else
			return 0;
	}
	else if (value == 3)
	{
		int number = stoi(data);
		if (this->middle_mark == number)
			return 1;
		else
			return 0;
	}
	else if (value == 4)
	{
		int number = stoi(data);
		if (this->prog_d == number)
			return 1;
		else
			return 0;
	}
	else if (value == 5)
	{
		int number = stoi(data);
		if (this->number_stud == number)
			return 1;
		else
			return 0;
	}
	else if (value == 6)
	{
		int number = 0;
		if (data == "true" || data == "true" || data == "1")
			number = 1;
		else
			number = 0;

		if (this->debt == number)
			return 1;
		else
			return 0;
	}
	else if (value == 7)
	{
		int number = stoi(data);
		if (this->course == number)
			return 1;
		else
			return 0;
	}
}
catch (const std::exception & ex)
{
	cout << ex.what() << endl;
	return 0;
}
```

```
	return 0;
}

bool Course::elementOutput(int value, string data)
{
	try
	{
		if (value == 1)
		{
			if (this->name == data)
				cout << *this << endl;
			return true;
		}
		else if (value == 2)
		{
			int number = stoi(data);
			if (this->age == number)
				cout << *this << endl;
			return true;
		}
		else if (value == 3)
		{
			int number = stoi(data);
			if (this->middle_mark == number)
				cout << *this << endl;
			return true;
		}
		else if (value == 4)
		{
			int number = stoi(data);
			if (this->prog_d == number)
				cout << *this << endl;
			return true;
		}
		else if (value == 5)
		{
			int number = stoi(data);
			if (this->number_stud == number)
				cout << *this << endl;
			return true;
		}
		else if (value == 6)
		{
			int number = 0;
			if (data == "true" || data == "true" || data == "1")
				number = 1;
			else
				number = 0;

			if (this->debt == number)
				return 1;
			else
				return 0;
		}
		else if (value == 7)
		{
			int number = stoi(data);
			if (this->course == number)
				cout << *this << endl;
			return true;
		}
	}
	catch (const std::exception & ex)
	{
```

```
			cout << ex.what() << endl;
			return 0;
		}

		return 0;
}

Course::Course(int a, int n, int m, string na, bool d, int pd, int c) : Student(a, n, m,
na, d, pd), course(c) {}
Course::Course() : Student(), course(1) {}
Course::Course(const Course& other) : Student(other), course(other.course) {}
Course::~Course() {}

bool Course::operator==(const int ns) const
{
		return this->number_stud == ns;
}
```

Functor.cpp

```
#include "Functor.h"

bool Functor::operator() (const shared_ptr<Student>& st1, const shared_ptr<Student>& st2)
{
		if (value % 2 != 0)
			return st1->get_numb() < st2->get_numb();
		else
			return st1->get_numb() > st2->get_numb();
}

Functor::Functor(int value) :value(value) {}
Functor::~Functor() {}
```

main.cpp

```
#include "class2.h"
#include "SmartPointer.h"

void func();

int main()
{
		setlocale(LC_ALL, "ru");

		func();

		if (_CrtDumpMemoryLeaks())
			cout << endl << "Обнаружена утечка!" << endl;
		else
			cout << endl << "Утечка памяти отсутствует." << endl;

		system("PAUSE");
		return 0;
}

void func()
{
		int value;
		vector <Student*> vector;

		try
		{
```

```
		auto_ptr<Course> autoptr(new Course);
		shared_ptr<Student> sharedptr = make_shared<Student>(19, 4, 9, "Jhon", 1,
14);
		unique_ptr<Course> uniqueptr = make_unique<Course>(17, 4, 10, "Peter", 0,
0, 1);
		SmartPointer<Course> MyPointer = new Course;
		weak_ptr<Student> weakptr = sharedptr;
		shared_ptr<Student> sharedptr2 = weakptr.lock();
		SmartPointer<Course> MyPointer2 = MyPointer;

		cout << endl << endl;
		cout << "Адрес объекта(умным указатель): " << MyPointer.operator->() <<
endl;
		cout << "Адрес копии объекта: " << MyPointer2.operator->() << endl;
		cout << endl;

		cout << setw(10) << "Возраст" << setw(8) << "Номер";
		cout << setw(15) << "Средний балл" << setw(7) << "Имя";
		cout << setw(10) << "Долг" << setw(14) << "Долг(прог.)";
		cout << setw(7) << "Курс" << endl;

		vector.emplace_back(autoptr.get());
		vector.emplace_back(sharedptr.get());
		vector.emplace_back(uniqueptr.get());
		vector.emplace_back(MyPointer.operator->());
		vector.emplace_back(sharedptr2.get());

		value = 1;
		for_each(vector.begin(), vector.end(), [&value](const Student* program)
		{
			cout << value << ". " << *program << endl;
			value++;
		});
		cout << endl;
	}
	catch (const std::exception& ex)
	{
		cout << ex.what() << endl << endl;
	}
}
```

4 Результати тестування

```
Создан умный указатель. Кол-во ссылок : 1
Умный указатель скопирован. Кол-во ссылок: 2

Адрес объекта(умным указатель): 00BA4548
Адрес копии объекта: 00BA4548

   Возраст   Номер   Средний балл    Имя      Долг   Долг(прог.)   Курс
1. 17        0       8               Bond     1      15            1
2. 19        4       9               Jhon     1      14
3. 17        4       10              Peter    0      0             1
4. 17        0       8               Bond     1      15            1
5. 19        4       9               Jhon     1      14

Умный указатель уничтожен. Кол-во ссылок: 1
Умный указатель уничтожен. Кол-во ссылок: 0

Утечка памяти отсутствует.
Press any key to continue . . .
```

5 Опис результатів

По результатах практичної роботи порівняли розумні вказівники бібліотеки STL.