Автор: Татаренко А., КІТ-119а

Дата: 17 червня 2020

Лабораторна робота №16.

РОБОТА З ДИНАМІЧНОЮ ПАМ’ЯТТЮ

Тема. Системна робота з динамічною пам’яттю.

Мета – дослідити особливості мови С++ при роботі з динамічною пам’яттю.

1 Завдання до роботи

Індивідуальне завдання 19.

Маючи класи з прикладної області РЗ (тільки базовий клас та клас / класи-спадкоємці), перевантажити оператори new / new [] та delete / delete []. Продемонструвати їх роботу і роботу операторів розміщення new / delete при розробці власного менеджера пам’яті (сховища).

Детальна інформація про власне сховище: є статично виділений масив заданого обсягу. Організувати виділення і звільнення пам’яті елементів ієрархії класів тільки у рамках цього сховища.

2 Розробка алгоритму розв’язання задачі.

2.1 Опис змінних

Arr stud_array; class Student; class Arr;

Класи, методи, функції, конструктори

3 Код програми

Course.h

```
#pragma once

#include "Student.h"

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

public:
	int get_age() const override;
	int get_number() const override;
	int get_mark() const override;
	int get_dprog()const override;
	bool get_debt()const override;
	string get_name() const override;

	string getInfo() const override;
	void enter(istream&) override;

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

	void* operator new(size_t);
	void* operator new[](size_t);
	void operator delete(void*);
	void operator delete[](void*);
};
```

Student.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 <regex>
#include <memory>
#include <vector>
#include <exception>
#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::istream;
using std::ostream;
using std::ofstream;
using std::stringstream;
using std::istringstream;
using std::regex;
using std::regex_match;
using std::regex_search;
using std::regex_replace;
using std::cmatch;
using std::unique_ptr;
using std::vector;
using std::exception;
using std::iterator;

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

public:
	virtual int get_age() const;
	virtual int get_number() const;
	virtual int get_mark() const;
	virtual int get_dprog()const;
	virtual bool get_debt()const;
	virtual string get_name() const;

	virtual string getInfo() const;
	virtual void enter(istream&);

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

	friend ofstream& operator<< (ofstream&, const Student&);
	friend ostream& operator<< (ostream&, const Student&);

	void* operator new(size_t);
	void* operator new[](size_t);
	void operator delete(void*);
	void operator delete[](void*);

};
```

Surnames.h

```
#pragma once

#include "Student.h"

class Surnames final : public Student
{
private:
	string star;
	string cur;
```

```
public:
	int get_age() const override;
	int get_number() const override;
	int get_mark() const override;
	int get_dprog()const override;
	bool get_debt()const override;
	string get_name() const override;

	string getInfo() const override;
	void enter(istream&) override;

	Surnames();
	Surnames(int, int, int, string, bool, int, string, string);
	Surnames(const Surnames&);
	~Surnames() override;

	void* operator new(size_t);
	void* operator new[](size_t);
	void operator delete(void*);
	void operator delete[](void*);
};
```

Course.cpp

```
#include "Course.h"
#include "Student.h"

int Course::get_age() const
{
	return age;
}
int Course::get_number() const
{
	return number_stud;
}
int Course::get_mark() const
{
	return middle_mark;
}
int Course::get_dprog() const
{
	return prog_d;
}
bool Course::get_debt() const
{
	return debt;
}
string Course::get_name() const
{
	return name;
}

string Course::getInfo() 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();
}
```

```
void Course::enter(istream& data)
{
	data >> age >> number_stud >> middle_mark >> prog_d >> debt >> name >> course;
}

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() {}

void* Course::operator new(size_t value)
{
	cout << "Вызов оператора new для класса Course" << endl;
	return ::operator new(value);
}

void* Course::operator new[](size_t value)
{
	cout << "Вызов оператора new[] для класса Course" << endl;
	return ::operator new[](value);
}

void Course::operator delete(void* pointer)
{
	cout << "Вызов оператора delete для класса Course" << endl;
	::operator delete(pointer);
}

void Course::operator delete[](void* pointer)
{
	cout << "Вызов оператора delete[] для класса Course" << endl;
	::operator delete(pointer);
}
```

main.cpp

```
#include "Surnames.h"
#include "Course.h"

#define SIZE 4

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

	Student* list[SIZE];

	list[0] = new Student();
	list[1] = new Course(18, 2, 8, "Andry", 1, 20, 2);
	list[2] = new Student(19, 3, 7, "Jhon", 1, 30);
	list[3] = new Surnames(19, 4, 10, "Dmitry", 0, 0, "Miler", "Grand");

	cout << endl;
	for (size_t i = 0; i < SIZE; i++)
		cout << *list[i] << endl;
	cout << endl;

	Student* list2 = new Student[SIZE];

	cout << endl;
	for (size_t i = 0; i < SIZE; i++)
		cout << list2[i] << endl;
```

```
	cout << endl;

	for (size_t i = 0; i < SIZE; i++)
		delete list[i];

	delete[] list2;

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

	system("PAUSE");

	return 0;
}
```

Student.cpp

```
#include "Student.h"

int Student::get_age() const
{
	return age;
}
int Student::get_number() const
{
	return number_stud;
}
int Student::get_mark() const
{
	return middle_mark;
}
int Student::get_dprog() const
{
	return prog_d;
}
bool Student::get_debt() const
{
	return debt;
}
string Student::get_name() const
{
	return name;
}

void Student::enter(istream& data)
{
	data >> age >> number_stud >> middle_mark >> prog_d >> debt >> name;
}

string Student::getInfo() 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;

	return temp.str();
}
```

```
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(1), 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";
}

ofstream& operator<< (ofstream& output, const Student& program)
{
	output << program.getInfo();
	return output;
}

ostream& operator<< (ostream& output, const Student& program)
{
	output << program.getInfo();
	return output;
}

void* Student::operator new(size_t value)
{
	cout << "Вызов оператора new для класса Student" << endl;
	return ::operator new(value);
}

void* Student::operator new[](size_t value)
{
	cout << "Вызов оператора new[] для класса Student" << endl;
	return ::operator new[](value);
}

void Student::operator delete(void* pointer)
{
	cout << "Вызов оператора delete для класса Student" << endl;
	::operator delete(pointer);
}

void Student::operator delete[](void* pointer)
{
	cout << "Вызов оператора delete[] для класса Student" << endl;
	::operator delete(pointer);
}
```

Surnames.cpp

```
#include "Surnames.h"

int Surnames::get_age() const
```

```
{
	return age;
}
int Surnames::get_number() const
{
	return number_stud;
}
int Surnames::get_mark() const
{
	return middle_mark;
}
int Surnames::get_dprog() const
{
	return prog_d;
}
bool Surnames::get_debt() const
{
	return debt;
}
string Surnames::get_name() const
{
	return name;
}

string Surnames::getInfo() 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(10) << star <<
setw(10) << cur;

	return temp.str();
}
void Surnames::enter(istream& data)
{
	data >> age >> number_stud >> middle_mark >> prog_d >> debt >> name >> star >>
cur;
}

Surnames::Surnames(int a, int n, int m, string na, bool d, int pd, string st, string cu)
: Student(a, n, m, na, d, pd), star(st), cur(cu) {}
Surnames::Surnames() : Student(), star("Petrov"), cur("Viktorovich") {}
Surnames::Surnames(const Surnames& other) : Student(other), star(other.star),
cur(other.cur) {}
Surnames::~Surnames() {}

void* Surnames::operator new(size_t value)
{
	cout << "Вызов оператора new для класса Surnames" << endl;
	return ::operator new(value);
}
void* Surnames::operator new[](size_t value)
{
	cout << "Вызов оператора new[] для класса Surnames" << endl;
	return ::operator new[](value);
}
void Surnames::operator delete(void* pointer)
{
	cout << "Вызов оператора delete для класса Surnames" << endl;
	::operator delete(pointer);
}
void Surnames::operator delete[](void* pointer)
```

```
{
	cout << "Вызов оператора delete[] для класса Surnames" << endl;
	::operator delete(pointer);
}
```

4 Результати тестування

```
Вызов оператора new для класса Student
Вызов оператора new для класса Course
Вызов оператора new для класса Student
Вызов оператора new для класса Surnames

17        1       8               Bond    1       15
18        2       8               Andry   1       20              2
19        3       7               Jhon    1       30
19        4       10              Dmitry  0       0               Miler     Grand

Вызов оператора new[] для класса Student

17        1       8               Bond    1       15
17        1       8               Bond    1       15
17        1       8               Bond    1       15
17        1       8               Bond    1       15

Вызов оператора delete для класса Student
Вызов оператора delete для класса Course
Вызов оператора delete для класса Student
Вызов оператора delete для класса Surnames
Вызов оператора delete[] для класса Student

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

5 Опис результатів

Дослідили особливості мови C++ при роботі з динамічною пам’яттю.