|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет  имени Н. Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н. Э. Баумана)** |

|  |  |
| --- | --- |
| ФАКУЛЬТЕТ | «Информатика, искусственный интеллект и системы управления» (ИУ) |

|  |  |
| --- | --- |
| КАФЕДРА | «Информационная безопасность» (ИУ8) |

Лабораторная работа № 8

ПО КУРСУ

«Алгоритмические языки»

# на тему «Изучение «умных» указателей»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Студент | ИУ8-24 |  |  |  | Соболева Д.Е. |
|  | (Группа) |  |  |  | (И. О. Фамилия) |
|  |  |  |  |  |  |
| Преподаватель: |  |  |  |  | Барыкин Д. В |
|  |  |  |  |  | (И.О. Фамилия) |

**Цель работы:**

Изучить умные указатели и научиться работать с ними.

**Текст программы:**

**1.**

#include <iostream>

using namespace std;

template<typename T>

class MyUnique {

private:

T\* ptr;

public:

MyUnique (T\* p = nullptr) : ptr(p) {}

~ MyUnique() {

delete ptr;

}

MyUnique (const MyUnique & other) = delete;

MyUnique & operator=(const MyUnique& other) = delete;

MyUnique(MyUnique&& other) {

ptr = other.ptr;

other.ptr = nullptr;

}

MyUnique& operator=(MyUnique&& other) {

if (this != &other) {

delete ptr;

ptr = other.ptr;

other.ptr = nullptr;

}

return \*this;

}

T\* get() const {

return ptr;

}

T& operator\*() const {

return \*ptr;

}

T\* operator->() const {

return ptr;

}

};

int main() {

MyUnique<int> ptr1(new int(42));

MyUnique<int> ptr2(move(ptr1));

if (ptr1.get() == nullptr) {

cout << "ptr1 is now empty\n";

}

\*ptr2 = 100;

cout << "\*ptr2: " << \*ptr2 << endl;

return 0;

}

**2.**

#include <iostream>

using namespace std;

template<typename T>

class MyShared {

private:

T\* ptr;

int\* count;

public:

MyShared(T\* ptr) : ptr(ptr), count(new int(1)) {}

MyShared(const MyShared& other) : ptr(other.ptr), count(other.count) {

(\*count)++;

}

~MyShared() {

(\*count)--;

if ((\*count) == 0) {

delete ptr;

delete count;

cout << "Memory freed!" << endl;

}

}

T& operator\*() {

return \*ptr;

}

T\* operator->() {

return ptr;

}

};

template<typename T, typename... Args>

MyShared<T> Make\_MyShared(Args&&... args) {

return MyShared<T>(new T(std::forward<Args>(args)...));

}

class MyClass {

public:

void hello() {

cout << "Hello from MyClass!" << endl;

}

};

int main() {

auto ptr1 = Make\_MyShared<MyClass>();

auto ptr2 = ptr1;

cout << "Using ptr1: ";

ptr1->hello();

cout << "Using ptr2: ";

ptr2->hello();

return 0;

}

**Вывод:**

Выполнив лабораторную №8, я изучила тему «умные» указатели и улучшила свои навыки в этой теме.