

## Лабораторна робота 15. РОЗУМНІ ВКАЗІВНИКИ

**Тема:** Розумні вказівники.

**Мета:** по результатах практичної роботи порівняти розумні вказівники бібліотеки STL.

### 1. Завдання до роботи

*Загальне завдання.* Створити STL-контейнер, що містить у собі об'єкти ієрархії класів, використати розумні вказівники:

- `auto_ptr`;
- `unique_ptr`;
- `shared_ptr`;
- `weak_ptr`.

*Додаткове завдання на оцінку «відмінно».* Створити власний розумний вказівник, поданий у вигляді шаблонного класу, який:

- має перевантажений оператор `*` та `->` для отримання фактичного об'єкта та його вказівника;
- дозволяє підраховувати кількість вказівників на об'єкт.

Продемонструвати дії, коли виникає інкремент та декремент кількості вказівників;

- контролювати виток пам'яті при виникненні виняткової ситуації.

*Додаткові умови виконання завдання:*

- продемонструвати відсутність витоків пам'яті при відсутності явного видалення пам'яті за допомогою функцій `delete` / `free`.

### 2.1. Опис класів

Базовий клас: `CProgram`.

Клас-спадкоємець: `CMalware`.

Клас-функтор: `Functor`.

Клас, який записує кількість посилань у вказівника: `PointerCount`.

Клас у якому реалізований функціонал свого розумного вказівника: `SmartPointer`.

### 2.2. Опис змінних

`int` `timeOfWork` – час роботи програми (змінна класу `CProgram`).

`int` `size` – розмір програми (змінна класу `CProgram`).

`int` `amountOfLines` – кількість рядків коду програми (змінна класу `CProgram`).

`int` `index` – номер програми (змінна класу `CProgram`).

`bool` useInternet – потребує програма Інтернет чи ні (змінна класу `CProgram`).  
`string` name – назва програми (змінна класу `CProgram`).  
`string` type – тип зловмисного ПО (змінна класу `CMalware`).

### 2.3. Опис методів

`virtual string` getInfo() `const` – виведення даних елемента у консоль (метод класу `CProgram`).  
`virtual stringstream` getStr() `const` – отримання строки з даними елемента (метод класу `CProgram`).  
`int` getID() `const` – отримання індекса елемента (метод класу `CProgram`).  
`bool` elementOutput(`int`, `string`) – виведення елемента за обраним критерієм (метод класу `CProgram`).  
`int` countElement(`int`, `string`) – виведення кількості елементів за обраним критерієм (метод класу `CProgram`).  
`CProgram()` – конструктор класа за замовчуванням (метод класу `CProgram`).  
`CProgram(bool, int, int, int, int, string)` – конструктор класа з параметрами (метод класу `CProgram`).  
`CProgram(const CProgram&)` – конструктор копіювання (метод класу `CProgram`).  
`virtual ~CProgram()` – деструктор класа (метод класу `CProgram`).  
`friend ostream& operator<< (ostream&, const CProgram&)` – перевантаження оператора `<<` (метод класу `CProgram`).  
`virtual bool operator==(const int) const` – перевантаження оператора `==` (метод класу `CProgram`).  
`bool operator()(const shared_ptr<CProgram>&, const shared_ptr<CProgram>&)` – перевантаження оператора `()` (метод класу `Functor`).

### 3. Текст програми

#### main.cpp

```
#include "malware.h"
#include "SmartPointer.h"
void func();

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

    func();

    if (_CrtDumpMemoryLeaks())
        cout << endl << "АХТУНГ!!!! ЕСТЬ УТЕЧКА ПАМЯТИ." << endl;
    else
        cout << endl << "Утечка памяти отсутствует." << endl;

    return 0;
}

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

    try
    {
```

```

        auto_ptr<CMalware> autoptr(new CMalware);
        shared_ptr<CProgram> sharedptr = make_shared<CProgram>(0, 423, 523, 654, 53453,
"Calculator");
        unique_ptr<CMalware> uniqueptr = make_unique<CMalware>(1, 8800, 555, 35, 35634,
"BestMalware", "Exploit");
        SmartPointer<CMalware> MyPointer = new CMalware;
        weak_ptr<CProgram> weakptr = sharedptr;
        shared_ptr<CProgram> sharedptr2 = weakptr.lock();
        SmartPointer<CMalware> MyPointer2 = MyPointer;

        cout << endl << endl;
        cout << "Адрес объекта, созданного умным указателем: " << MyPointer.operator->() <<
endl;

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

        cout << setw(12) << "Название" << setw(14) << "Индекс";
        cout << setw(14) << "Время работы" << setw(8) << "Размер";
        cout << setw(18) << "Количество линий" << setw(10) << "Интернет";
        cout << setw(10) << "Тип" << 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 CProgram* program)
        {
            cout << value << ". " << *program << endl;
            value++;
        });
        cout << endl;
    }
    catch (const std::exception& ex)
    {
        cout << ex.what() << endl << endl;
    }
}

```

## malware.cpp

```

#include "malware.h"
stringstream CMalware::getStr() const
{
    stringstream temp;

    temp << name << " " << index << " " << timeOfWork
        << " " << size << " " << amountOfLines << " "
        << useInternet << " " << type;

    return temp;
}
string CMalware::getInfo() const
{
    stringstream temp;

    temp.setf(ios::left);
    temp << setw(18) << name << setw(12) << index
        << setw(11) << timeOfWork << setw(13) << size
        << setw(12) << amountOfLines << setw(12) << boolalpha << useInternet
        << setw(14) << type;

    return temp.str();
}
int CMalware::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->timeOfWork == number)
                return 1;
            else
                return 0;
        }
        else if (value == 3)
        {
            int number = stoi(data);
            if (this->size == number)
                return 1;
            else
                return 0;
        }
        else if (value == 4)
        {
            int number = stoi(data);
            if (this->amountOfLines == number)
                return 1;
            else
                return 0;
        }
        else if (value == 5)
        {
            int number = stoi(data);
            if (this->index == 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->useInternet == number)
                return 1;
            else
                return 0;
        }
        else if (value == 7)
        {
            if (this->type == data)
                return 1;
            else
                return 0;
        }
    }
    catch (const std::exception & ex)
    {
        cout << ex.what() << endl;
        return 0;
    }
}

```

```

        return 0;
    }
    bool CMalware::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->timeOfWork == number)
                    cout << *this << endl;
                return true;
            }
            else if (value == 3)
            {
                int number = stoi(data);
                if (this->size == number)
                    cout << *this << endl;
                return true;
            }
            else if (value == 4)
            {
                int number = stoi(data);
                if (this->amountOfLines == number)
                    cout << *this << endl;
                return true;
            }
            else if (value == 5)
            {
                int number = stoi(data);
                if (this->index == 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->useInternet == number)
                    return 1;
                else
                    return 0;
            }
            else if (value == 7)
            {
                if (this->type == data)
                    cout << *this << endl;
                return true;
            }
        }
        catch (const std::exception & ex)
        {
            cout << ex.what() << endl;
            return 0;
        }

        return 0;
    }
}

```

```

CMalware::CMalware(bool internet, int time, int size, int lines, int index, string name, string
type) : CProgram(internet, time, size, lines, index, name), type(type) {}
CMalware::CMalware() : CProgram(), type("Exploit") {}
CMalware::CMalware(const CMalware& other) : CProgram(other), type(other.type) {}
CMalware::~CMalware() {}

bool CMalware::operator==(const int id) const
{
    return this->index == id;
}

```

## program.cpp

```

#include "program.h"

string CProgram::getInfo() const
{
    stringstream temp;

    temp.setf(std::ios::left);
    temp << setw(18) << name << setw(12) << index << setw(11)
        << timeOfWork << setw(13) << size << setw(12)
        << amountOfLines << setw(8) << boolalpha << useInternet;

    return temp.str();
}
int CProgram::getID() const
{
    return index;
}
stringstream CProgram::getStr() const
{
    stringstream temp;
    temp << name << " " << index << " " << timeOfWork << " "
        << size << " " << amountOfLines << " " << useInternet;

    return temp;
}
int CProgram::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->timeOfWork == number)
                return 1;
            else
                return 0;
        }
        else if (value == 3)
        {
            int number = stoi(data);
            if (this->size == number)
                return 1;
            else
                return 0;
        }
        else if (value == 4)
        {

```

```

        int number = stoi(data);
        if (this->amountOfLines == number)
            return 1;
        else
            return 0;
    }
    else if (value == 5)
    {
        int number = stoi(data);
        if (this->index == 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->useInternet == number)
            return 1;
        else
            return 0;
    }
}
catch (const std::exception& ex)
{
    cout << ex.what() << endl;
    return 0;
}

return 0;
}

bool CProgram::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->timeOfWork == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 3)
        {
            int number = stoi(data);
            if (this->size == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 4)
        {
            int number = stoi(data);
            if (this->amountOfLines == number)
                cout << *this << endl;
            return true;
        }
        else if (value == 5)
        {

```

```

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

return 0;
}

ostream& operator<< (ostream& output, const CProgram& program)
{
    output << program.getInfo();
    return output;
}
bool CProgram::operator==(const int id) const
{
    return this->index == id;
}

CProgram::CProgram(bool internet, int time, int size, int lines, int index, string name) :
useInternet(internet), timeOfWork(time), size(size), amountOfLines(lines), index(index), name(name)
{}
CProgram::CProgram() : useInternet(false), timeOfWork(0), size(0), amountOfLines(0), index(0101),
name("Basic") {}
CProgram::CProgram(const CProgram& other) : useInternet(other.useInternet),
timeOfWork(other.timeOfWork), size(other.size), amountOfLines(other.amountOfLines),
index(other.index), name(other.name) {}
CProgram::~~CProgram() {}

```

## 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;

```

## malware.h

```

#pragma once
#include "program.h"
class CMalware final: public CProgram
{
private:
    string type;

public:
    string getInfo() const override final;
    stringstream getStr() const override final;
    bool elementOutput(int, string) override final;
    int countElement(int, string) override final;

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

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

```

## program.h

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

class CProgram {
protected:
    int timeOfWork;           //average time of program execution
    int size;                 //size of program
    int amountOfLines;        //number of lines in code
    int index;                //index
    bool useInternet;         //use internet
    string name;              //name of program

public:
    virtual string getInfo() const;
    virtual stringstream getStr() const;
    int getID() const;
    virtual bool elementOutput(int, string);
    virtual int countElement(int, string);

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

    friend ostream& operator<< (ostream&, const CProgram&);
    virtual bool operator==(const int) const;
};
```

## Functor.cpp

```
#include "Functor.h"

bool Functor::operator() (const shared_ptr<CProgram>& program1, const shared_ptr<CProgram>&
program2)
{
    if (value % 2 != 0)
        return program1->getID() < program2->getID();
    else
        return program1->getID() > program2->getID();
}

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

## Functor.h

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

class Functor
{
private:
    int value;

public:
    bool operator()(const shared_ptr<CProgram>&, const shared_ptr<CProgram>&);
    Functor(int);
    ~Functor();
};
```

## Functor.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 собственного умного указателя. Количество 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 собственный умный указатель. Количество ссылок : " << pointerCount-
>GetPointerCount() << endl;
    }
    virtual ~SmartPointer()
    {

```

```

        if (pointerCount)
        {
            int decrementCount = pointerCount->Decrement();
            cout << "Собственный умный указатель уничтожен. Количество 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. Количество ссылок: " << pointerCount-
>GetPointerCount();
    }
};

```

## 4. Результати роботи програми

```

Создан собственный умный указатель. Количество ссылок : 1
Умный указатель скопирован. Количество ссылок: 2

Адрес объекта, созданного умным указателем: 0000018DF1469ED0
Адрес копии созданного объекта: 0000018DF1469ED0

    Название      Индекс  Время работы  Размер  Количество линий  Интернет  Тип
1. Basic         65      0             0       0                false     Exploit
2. Calculator    53453   423           523     654              false     Exploit
3. BestMalware   35634   8800          555     35               true      Exploit
4. Basic         65      0             0       0                false     Exploit
5. Calculator    53453   423           523     654              false     Exploit

Собственный умный указатель уничтожен. Количество ссылок: 1
Собственный умный указатель уничтожен. Количество ссылок: 0

Утечка памяти отсутствует.

```

## 5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з розумними вказівниками. Був створений власний розумний вказівник у шаблонному класі, який має перевантажені оператори \* та ->. Також був створений клас, який підраховує кількість вказівників на об'єкт.

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