

Звіт

Автор: Момот Р.Є. КІТ-119а

Дата: 27 травня 2020

Лабораторна робота 16. РОБОТА З ДИНАМІЧНОЮ ПАМ'ЯТТЮ

Тема: Системна робота з динамічною пам'яттю.

Мета: дослідити особливості мови C++ при роботі з динамічною пам'яттю.

1. Завдання до роботи

Загальне завдання. 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.

2.1. Опис класів

Базовий клас: `CProgram`.

Перший клас-спадкоємець: `CMalware`.

Другий клас-спадкоємець: `CProgramForSale`.

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`).
`int` `price` – ціна програми (змінна класу `CProgramForSale`).

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 класу за замовчуванням (метод класу `CProgram`).

`CProgram(bool, int, int, int, int, string)` – конструктор класу з параметрами (метод класу `CProgram`).

`CProgram(const CProgram&)` – конструктор копіювання (метод класу `CProgram`).

`virtual ~CProgram()` – деструктор класу (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`).

`void* operator new(size_t)` – перевантаження оператора `new`.

`void* operator new[](size_t)` – перевантаження оператора `new[]`.

`void operator delete(void*)` – перевантаження оператора `delete`.

`void operator delete[](void*)` – перевантаження оператора `delete[]`.

3. Текст програми

main.cpp

```
#include "programForSale.h"
#include "malware.h"
#define SIZE 4
int main()
{
    setlocale(LC_ALL, "Rus");

    CProgram* list[SIZE];

    list[0] = new CProgram();
    list[1] = new CMalware(1, 8800, 555, 35, 35634, "BestMalware", "Exploit");
    list[2] = new CProgram(0, 423, 523, 654, 53453, "Calculator");
    list[3] = new CProgramForSale(0, 345, 789, 423, 67456, "MoneyStealer", 54545);

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

    CProgram* list2 = new CProgram[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;

return 0;
}

```

malware.cpp

```

/**
 * @file malware.cpp
 * Файл реалізації методів класу-спадкоємця
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */

#include "malware.h"           /** Підключення файлу malware.h */
#include "program.h"

int CMalware::getTime() const    /** Реалізація геттера часу виконання програми */
{
    return timeOfWork;           /** Повернення часу виконання програми */
}
int CMalware::getSize() const    /** Реалізація геттера розміру програми */
{
    return size;                 /** Повернення розміру програми */
}
int CMalware::getLines() const   /** Реалізація геттера кількості рядків програми */
{
    return amountOfLines;        /** Повернення розміру програми */
}
int CMalware::getIndex() const   /** Реалізація геттера індексу програми */
{
    return index;                /** Повернення розміру програми */
}
bool CMalware::getInternet()const /** Реалізація геттера інтернету */
{
    return useInternet;          /** Повернення змінної інтернету */
}
string CMalware::getName()const  /** Реалізація геттера назви програми */
{
    return name;                 /** Повернення назви програми */
}

string CMalware::getInfo() const /** Реалізація функції отримання інформації */
{
    stringstream temp;           /** Оголошення змінної stringstream */
    temp.setf(ios::left);

    temp << setw(10) << timeOfWork << setw(12) << size    /** Отримання рядка з даними */
        << setw(9) << amountOfLines << setw(12) << boolalpha << useInternet
        << setw(11) << index << setw(20) << name
        << setw(14) << type;

    return temp.str();            /** Повернення рядка з інформацією */
}
void CMalware::enter(istream& data) /** Реалізація перевантаженого оператора вводу */
{
    data >> index >> timeOfWork >> type >> size >> amountOfLines >> useInternet >> name;    /**
Введення даних у об'єкт типу istream */
}

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) {} /** Реалізація
конструктора з параметрами */
CMalware::CMalware() : CProgram(), type("Trojan") {} /** Реалізація конструктора за замовчуванням
*/

```

```

CMalware::CMalware(const CMalware& other) : CProgram(other), type(other.type) {} /** Реалізація
конструктора копіювання */
CMalware::~CMalware() {} /** Реалізація деструктора */

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

```

program.cpp

```

/**
 * @file program.cpp
 * Файл реалізації методів базового класу
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */

#include "program.h"

int CProgram::getTime() const /** Реалізація геттера часу роботи програми */
{
    return timeOfWork; /** Повернення часу роботи програми */
}
int CProgram::getSize() const /** Реалізація геттера розміру програми */
{
    return size; /** Повернення розміру програми */
}
int CProgram::getLines() const /** Реалізація геттера кількості рядків коду програми */
{
    return amountOfLines; /** Повернення кількості рядків коду програми */
}
int CProgram::getIndex() const /** Реалізація геттера індексу програми */
{
    return index; /** Повернення індексу програми */
}
bool CProgram::getInternet()const /** Реалізація геттера інтернету */
{
    return useInternet; /** Повернення інтернету */
}
string CProgram::getName()const /** Реалізація геттера назви програми */
{
    return name; /** Повернення назви програми */
}

void CProgram::enter(istream& data) /** Реалізація перевантаженого оператора вводу */
{
    data >> index >> timeOfWork >> size >> amountOfLines >> useInternet >> name; /** Введення
даних у об'єкт типу istream */
}

```

```

string CProgram::getInfo() const  /** Реалізація методу отримання даних програми */
{
    stringstream temp;           /** Створення змінної типу stringstream */
    temp.setf(ios::left);

    temp << setw(10) << timeOfWork << setw(12) << size
        << setw(9) << amountOfLines << setw(12) << boolalpha << useInternet
        << setw(11) << index << setw(16) << name; /** Запис даних у об'єкт типу stringstream */

    return temp.str();           /** Повернення даних у форматі string */
}

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)
{}                               /** Реалізація конструктора з параметрами */
CProgram::CProgram() : useInternet(false), timeOfWork(200), size(200), amountOfLines(200),
index(0101), name("Basic") {}   /** Реалізація конструктора за замовчуванням */
CProgram::CProgram(const CProgram& other) : useInternet(other.useInternet),
timeOfWork(other.timeOfWork), size(other.size), amountOfLines(other.amountOfLines),
index(other.index), name(other.name) {} /** Реалізація конструктора копіювання */
CProgram::~CProgram() {}        /** Реалізація деструктора */

ofstream& operator<< (ofstream& output, const CProgram& program)  /** Реалізація перевантаженого
оператора запису даних у файл */
{
    output << program.getInfo();  /** Виклик фнкції отримання інформації */
    return output;               /** Повернення інформації */
}

ostream& operator<< (ostream& output, const CProgram& program)    /** Реалізація
перевантаженого оператора виводу даних у консоль */
{
    output << program.getInfo();  /** Виклик фнкції отримання інформації */
    return output;               /** Повернення інформації */
}

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

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

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

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

```

malware.h

```

/**
 * @file malware.h
 * Файл оголошення класу спадкоємця
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */

#include "program.h"

```

```

#pragma once

class CMalware final: public CProgram          /** Оголошення класу спадкоємця */
{
private:
    string type;                               /** Підключення файлу program.h */

public:
    int getTime() const override; /** Оголошення перевантаженого гетера часу виконання програми */
    int getSize() const override;  /** Оголошення перевантаженого гетера розміру програми */
    int getLines() const override; /** Оголошення перевантаженого гетера кількості рядків коду
програми */
    int getIndex() const override; /** Оголошення перевантаженого гетера індексу програми */
    bool getInternet() const override; /** Оголошення перевантаженого гетера Інтернету програми */
    string getName() const override; /** Оголошення перевантаженого гетера назви програми */

    string getInfo() const override; /** Оголошення перевантаженого метода отримання інформації
програми */
    void enter(istream&) override;  /** Оголошення перевантаженого метода введення інформації
програми */

    CMalware();                       /** Оголошення конструктора за замовчуванням */
    CMalware(bool, int, int, int, int, string, string); /** Оголошення конструктора з
параметрами */
    CMalware(const CMalware&);         /** Оголошення конструктора копіювання */
    ~CMalware() override;             /** Оголошення перевантаженого деструктора */

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

```

program.h

```

/**
 * @file program.h
 * Підключення необхідних бібліотек та оголошення класу CProgram.
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */

#pragma once

#define _CRT_SECURE_NO_WARNINGS
#define CRTDBG_MAP_ALLOC
#include <crtdbg.h> /** Підключення бібліотеки crtdbg.h */
#define DEBUG_NEW new(_NORMAL_BLOCK, FILE, __LINE__)

#include <string>      /** Підключення бібліотеки string */
#include <iostream>    /** Підключення бібліотеки iostream */
#include <iomanip>      /** Підключення бібліотеки iomanip */
#include <locale>      /** Підключення бібліотеки locale */
#include <fstream>     /** Підключення бібліотеки fstream */
#include <sstream>     /** Підключення бібліотеки sstream */
#include <regex>       /** Підключення бібліотеки regex */
#include <memory>      /** Підключення бібліотеки memory */
#include <vector>      /** Підключення бібліотеки vector */
#include <exception>   /** Підключення бібліотеки exception */
#include <iterator>    /** Підключення бібліотеки 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 CProgram {                                /** Оголошення базового класу*/
protected:
    int timeOfWork;                            /** Час виконання програми*/
    int size;                                  /** Розмір програми*/
    int amountOfLines;                         /** Кількість рядків коду програми*/
    int index;                                 /** Індекс програми*/
    bool useInternet;                          /** Використовує програма Інтернет чи ні*/
    string name;                              /** Назва програми*/

public:
    virtual int getTime() const;               /** Оголошення віртуального гетера отримання часу роботи
програми*/
    virtual int getSize() const;               /** Оголошення віртуального гетера отримання розміру програми*/
    virtual int getLines() const;              /** Оголошення віртуального гетера отримання кількості
рядків програми*/
    virtual int getIndex()const;               /** Оголошення віртуального гетера отримання індексу програми*/
    virtual bool getInternet()const;           /** Оголошення віртуального гетера отримання часу роботи
програми*/
    virtual string getName() const;            /** Оголошення віртуального гетера отримання назви програми*/

    virtual string getInfo() const;            /** Оголошення віртуальної функції отримання інформації
програми*/
    virtual void enter(istream&);

    CProgram();                               /** Оголошення конструктора по замовчуванням*/
    CProgram(bool, int, int, int, int, string); /** Оголошення конструктора з параметрами*/
    CProgram(const CProgram& other);           /** Оголошення конструктора копіювання*/
    virtual ~CProgram();                       /** Оголошення віртуального деструктора*/

    friend ostream& operator<< (ostream&,const CProgram&); /** Оголошення перевантаженого
оператора виводу у файл*/
    friend ostream& operator<< (ostream&, const CProgram&); /** Оголошення перевантаженого
оператора виводу у консоль*/

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

```

programForSale.h

```

/**
 * @file programForSale.h
 * Файл оголошення класу-спадкоємця.
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */

```



```

#pragma once
#include "program.h"          /** Підключення файлу program.h */
class CProgramForSale final : public CProgram /** Оголошення класу спадкоємця */
{
private:
    int price;                /** Ціна програми */

public:
    int getTime() const override; /** Оголошення перевантаженого гетера отримання часу
виконання програми */
    int getSize() const override; /** Оголошення перевантаженого гетера отримання
розміру програми */
    int getLines() const override; /** Оголошення перевантаженого гетера отримання
кількості рядків коду програми */
    int getIndex() const override; /** Оголошення перевантаженого гетера отримання
індексу програми */
    bool getInternet() const override; /** Оголошення перевантаженого гетера отримання часу
виконання програми */
    string getName() const override; /** Оголошення перевантаженого гетера отримання назви
програми */

    string getInfo() const override; /** Оголошення перевантаженого метода отримання інформації
програми */
    void enter(istream&) override; /** Оголошення перевантаженого метода вводу
інформації програми */

    CProgramForSale();          /** Оголошення конструктора за замовчуванням */
    CProgramForSale(bool, int, int, int, int, string, int); /** Оголошення конструктора з
параметрами */
    CProgramForSale(const CProgramForSale&); /** Оголошення конструктора копіювання */
    ~CProgramForSale() override; /** Оголошення перевантаженого деструктора */

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

```

programForSale.cpp

```

/**
 * @file programForSale.cpp
 * Файл реалізації методів класу-спадкоємця
 * @author Momot Roman
 * @version 1.0
 * @date 2020.05.26
 */

#include "programForSale.h" /** Підключення файлу programForSale.h */

int CProgramForSale::getTime() const /** Реалізація геттера часу роботи програми */
{
    return timeOfWork; /** Повернення часу роботи програми */
}
int CProgramForSale::getSize() const /** Реалізація геттера розміру програми */
{
    return size; /** Повернення розміру програми */
}
int CProgramForSale::getLines() const /** Реалізація геттера кількості рядків коду програми */
{
    return amountOfLines; /** Повернення кількості рядків коду програми */
}
int CProgramForSale::getIndex() const /** Реалізація геттера індексу програми */

```



```

{
    return index;                                /** Повернення індексу програми */
}
bool CProgramForSale::getInternet()const          /** Реалізація геттера інтернету */
{
    return useInternet;                          /** Повернення інтернету */
}
string CProgramForSale::getName()const            /** Реалізація геттера назви програми */
{
    return name;                                 /** Повернення назви програми */
}

string CProgramForSale::getInfo() const           /** Реалізація методу отримання даних програми */
{
    stringstream temp;                          /** Створення змінної типу stringstream */
    temp.setf(ios::left);

    temp << setw(10) << timeOfWork << setw(12) << size
        << setw(9) << amountOfLines << setw(12) << boolalpha << useInternet
        << setw(11) << index << setw(20) << name
        << setw(14) << price;                    /** Запис даних у об'єкт типу stringstream */

    return temp.str();                          /** Повернення даних у форматі string */
}
void CProgramForSale::enter(istream& data)        /** Реалізація перевантаженого оператора вводу */
{
    data >> index >> timeOfWork >> price >> size >> amountOfLines >> useInternet >> name; /**
Введення даних у об'єкт типу istream */
}

CProgramForSale::CProgramForSale(bool internet, int time, int size, int lines, int index, string
name, int price) : CProgram(internet, time, size, lines, index, name), price(price) {} /**
Реалізація конструктора з параметрами */
CProgramForSale::CProgramForSale() : CProgram(), price(2000) {}                      /** Реалізація
конструктора за замовчуванням */
CProgramForSale::CProgramForSale(const CProgramForSale& other) : CProgram(other), price(other.price)
{} /** Реалізація конструктора копіювання */
CProgramForSale::~CProgramForSale() {} /** Реалізація деструктора */

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

```

4. Результати роботи програми

```
Вызов оператора new для класса CProgram
Вызов оператора new для класса CMalware
Вызов оператора new для класса CProgram
Вызов оператора new для класса CProgramForSale

200      200      200      false      65      Basic
8800     555      35      true       35634    BestMalware      Exploit
423      523      654     false      53453    Calculator
345      789      423     false      67456    MoneyStealer     54545

Вызов оператора new[] для класса CProgram

200      200      200      false      65      Basic
200      200      200      false      65      Basic
200      200      200      false      65      Basic
200      200      200      false      65      Basic

Вызов оператора delete для класса CProgram
Вызов оператора delete для класса CMalware
Вызов оператора delete для класса CProgram
Вызов оператора delete для класса CProgramForSale
Вызов оператора delete[] для класса CProgram

Утечка памяти отсутствует.
```

5. Висновки

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з динамічною пам'яттю. В усіх класах були перевантажені оператори new, new[], delete та delete[].

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