Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення
Дисципліна: Об’єктно-орієнтоване програмування

Лабораторна робота №5
Тема: «ПЕРЕВАНТАЖЕННЯ ОПЕРАТОРІВ»

Виконав: ст. гр. КН-24
Куріщенко П. В.
Перевірив: асистент
Козірова Н. Л.

Кропивницький 2025

Мета: ознайомитись з поняттям перевантаження операторів та навчитись їх програмно реалізовувати мовою C++.

Варіант – 14 - 10 = 4

Завдання 1:

Перевантажте оператори для завдання № 2 з лабораторної роботи №2.

До перевантаження операторів << і >>:

```
#include <musicalband.h>

int main()
{
    string input;
    cout << "Enter band name: ";
    getline(cin, input);
    QString bandName = QString::fromStdString(input);

    cout << "Enter band style: ";
    getline(cin, input);
    QString bandStyle = QString::fromStdString(input);

    cout << "Enter band leader name: ";
    getline(cin, input);
    BandLeader leader(QString::fromStdString(input));

    cout << "Enter number of musicians: ";
    int numMusicians;
    cin >> numMusicians;
    cin.ignore();

    MusicalBand band(bandName, bandStyle, leader);

    for (int i = 0; i < numMusicians; ++i) {
        cout << "\n--- Musician " << (i+1) << " ---\n";

        cout << "Enter musician name: ";
        getline(cin, input);
        QString musicianName = QString::fromStdString(input);

        cout << "Enter instrument name: ";
        getline(cin, input);
        QString instrumentName = QString::fromStdString(input);

        cout << "Enter instrument production year: ";
        unsigned short year;
        cin >> year;
        cin.ignore();
        Instrument instrument(instrumentName, year);
```

```
        cout << "Enter manager name: ";
        getline(cin, input);
        Manager* manager = new Manager(QString::fromStdString(input));

        Musician* musician = new Musician(musicianName, instrument, manager);
        band.addMusician(musician);
    }

    cout << "\n--- Band Info ---\n";
    cout << "Band: " << band.getName().toStdString()
         << " (" << band.getStyle().toStdString() << ")" << endl;
    cout << "Leader: " << leader.getName().toStdString() << endl;

    for (int i = 0; i < band.getLen(); ++i) {
        Musician* m = band.getList().at(i);
        cout << "Musician " << (i+1) << ": " << m->getName().toStdString() <<
endl;
    }

    for (Musician* m : band.getList()) {
        band.delMusician(m);
    }

    return 0;
}
```

Після перевантаження операторів << і >>:

```
#include <musicalband.h>

int main()
{
    MusicalBand band;

    cin >> band;
    cout << "\n--- Band Info ---\n";
    cout << band;

    for (Musician* m : band.getList()) {
        band.delMusician(m);
    }

    return 0;
}
```

Лістинг **musicalband.cpp**:

```
istream& operator>>(istream& in, MusicalBand& band) {
    string input;
    cout << "Enter band name: ";
    getline(in, input);
    band.name = QString::fromStdString(input);

    cout << "Enter band style: ";
    getline(in, input);
    band.style = QString::fromStdString(input);
```

```
    in >> band.leader;

    cout << "Enter number of musicians: ";
    int numMusicians;
    in >> numMusicians;
    in.ignore();

    for (int i = 0; i < numMusicians; ++i) {
        cout << "\n--- Musician " << (i + 1) << " ---\n";
        Musician* m = new Musician();
        in >> *m;
        band.addMusician(m);
    }

    return in;
}

ostream& operator<<(ostream& out, const MusicalBand& band) {
    out << "Band: " << band.name.toStdString()
    << " (" << band.style.toStdString() << ")\n"
    << band.leader << "\n";

    int i = 1;
    for (auto* m : band.musicians)
        out << "  Musician " << i++ << ": " << *m << "\n";

    return out;
}
```

Лістинг **musician.cpp**:

```
istream& operator>>(istream& in, Musician& musician) {
    string input;

    cout << "Enter musician name: ";
    getline(in, input);
    musician.name = QString::fromStdString(input);

    in >> musician.instrument;

    musician.manager = new Manager();
    in >> *musician.manager;

    return in;
}

ostream& operator<<(ostream& out, const Musician& musician) {
    out << "Musician: " << musician.getName().toStdString()
    << " | ";
    out << musician.getInstrument();
    out << " | ";
    out << (musician.getManager() ? *musician.getManager() :
Manager("None"));
    return out;
}
```

Лістинг **bandleader.cpp**:

```
istream& operator>>(istream& in, BandLeader& leader) {
    string input;
    cout << "Enter band leader name: ";
    getline(in, input);
    leader.name = QString::fromStdString(input);
    return in;
}

ostream& operator<<(ostream& out, const BandLeader& leader) {
    out << "Leader: " << leader.name.toStdString();
    return out;
}
```

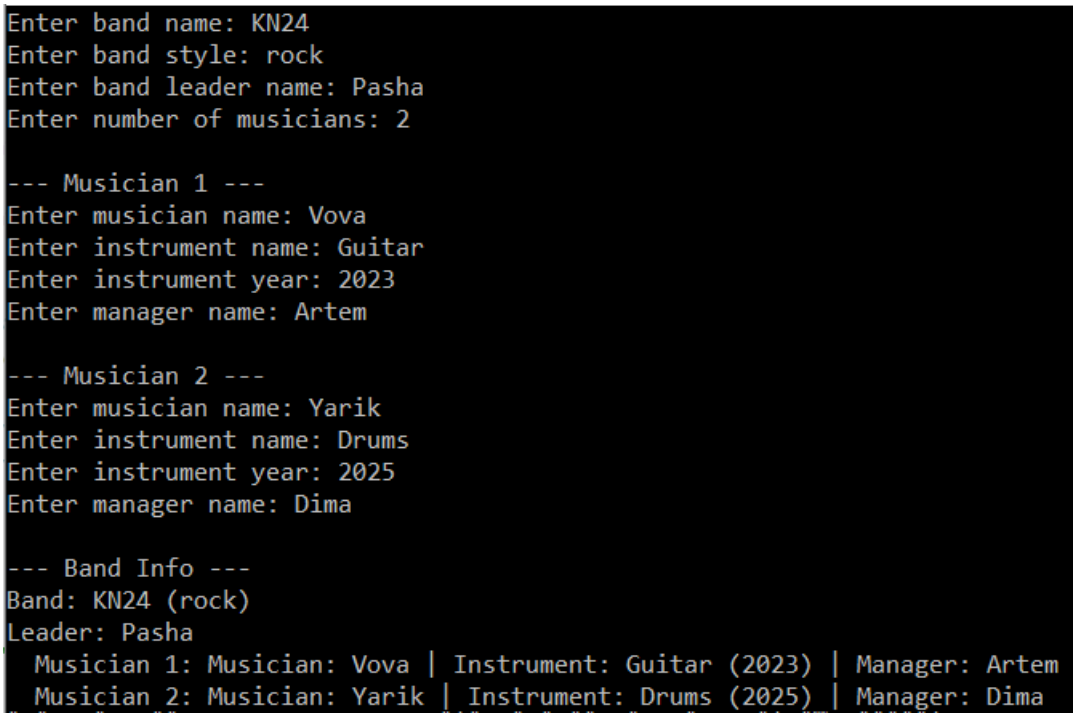
Лістинг **instrument.cpp**:

```
istream& operator>>(istream& in, Instrument& instrument) {
    string input;

    cout << "Enter instrument name: ";
    getline(in, input);
    instrument.name = QString::fromStdString(input);

    cout << "Enter production year: ";
    unsigned short year;
    in >> year;
    in.ignore();
    instrument.prodYear = year;

    return in;
}

ostream& operator<<(ostream& out, const Instrument& instrument) {
    out << instrument.name.toStdString() << " (" << instrument.prodYear <<
")";
    return out;
}
```

Лістинг **manager.cpp**:

```
istream& operator>>(istream& in, Manager& manager) {
    string input;
    cout << "Enter manager name: ";
    getline(in, input);
    manager.name = QString::fromStdString(input);
    return in;
}

ostream& operator<<(ostream& out, const Manager& manager) {
    out << "Manager: " << manager.name.toStdString();
    return out;
}
```

Результат:

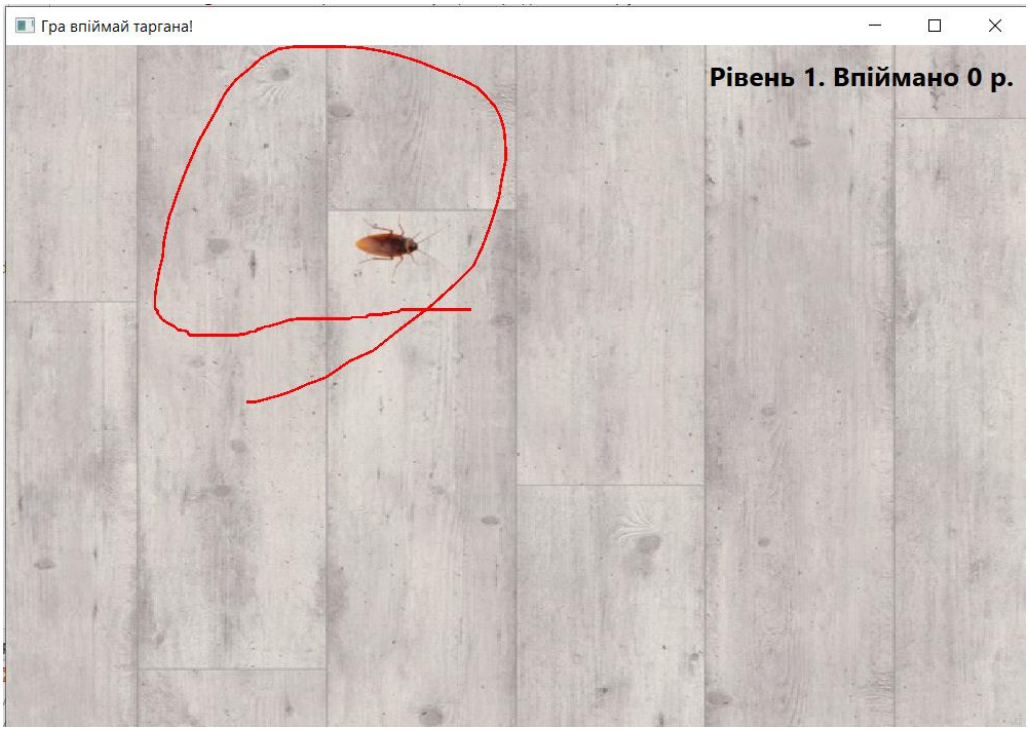


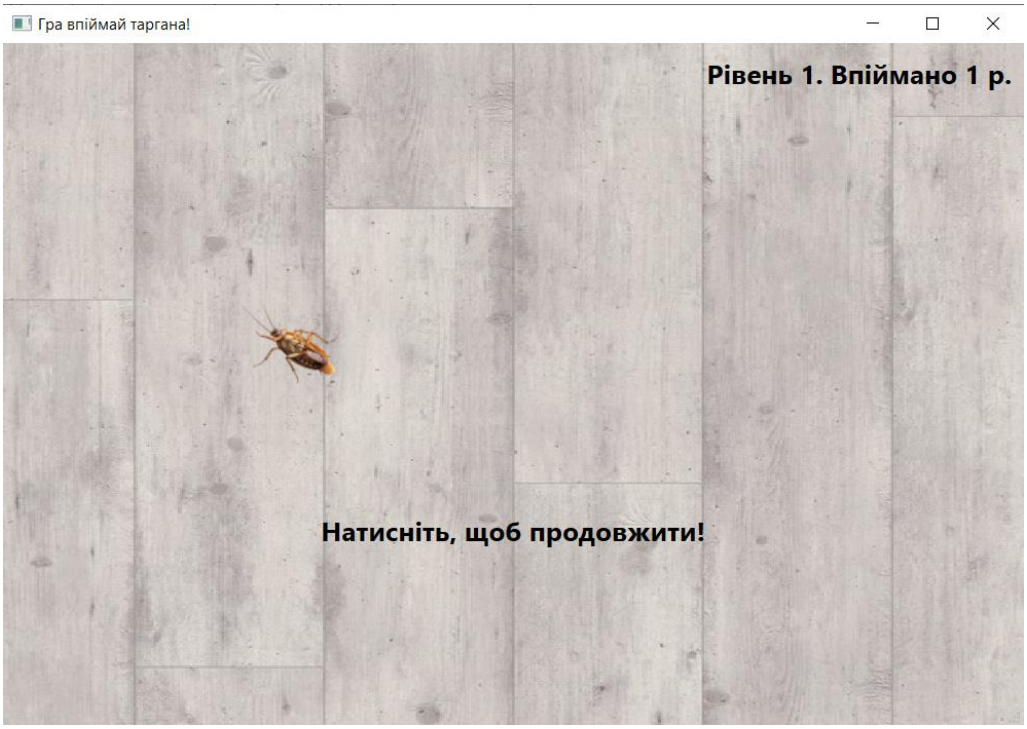
Завдання 2:

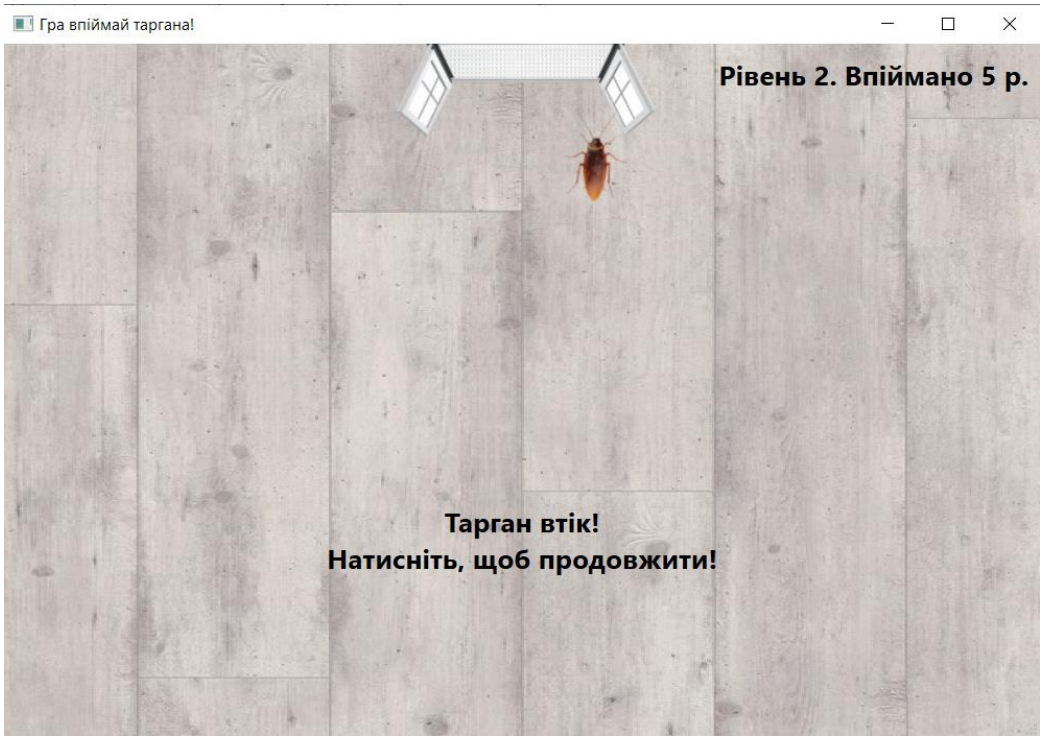
Продовжити розробку гри «Арештуй таргана». Тепер ваш тарган має стати окремим класом. Реалізуйте кілька рівнів гри — на кожному наступному рівні має з’являтися новий тарган, який буде класом-нащадком. Починаючи з другого рівня, на формі має з’явитися вікно-втечі, у яке тарган намагатиметься втекти. Завдання гравця — відігнати таргана від вікна та спіймати його, обвівши мишею.

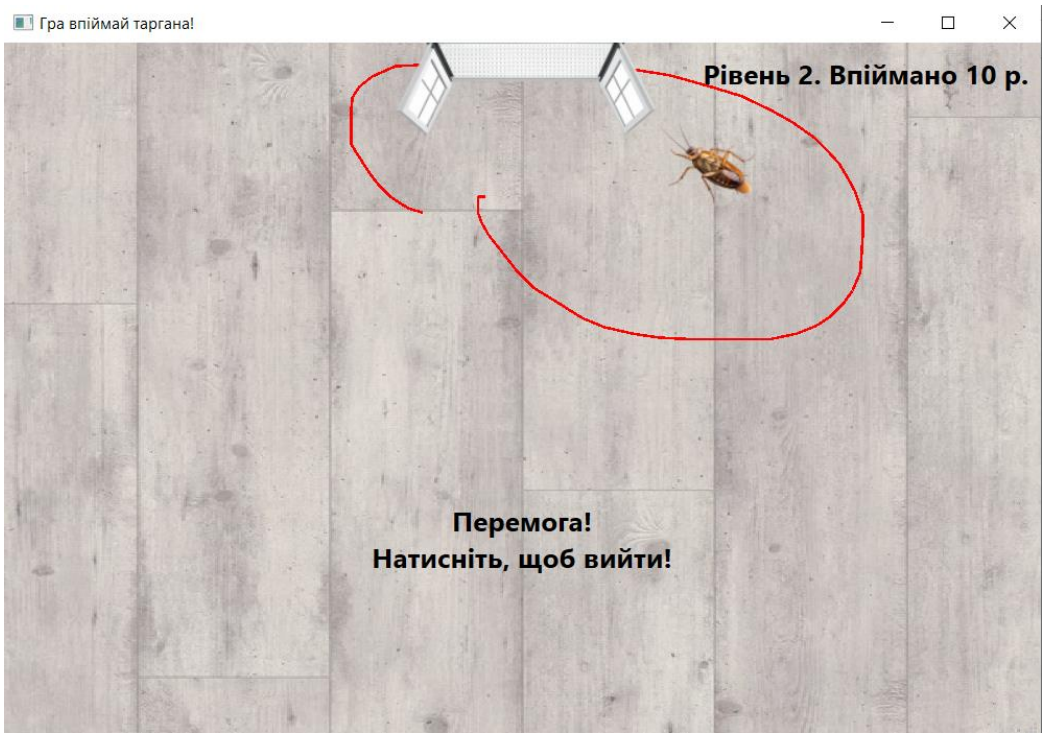
Лістинг (github)

Результат:


Гра впіймай таргана!
Рівень 1. Впіймано 0 р.


Гра впіймай таргана!
Рівень 1. Впіймано 1 р.
Натисніть, щоб продовжити!





Завдання 3:

Варіант 4

Створіть клас String, який представляє рядок символів. В класі String перевантажте наступні оператори:

1. Оператор + для конкатенації двох рядків.
2. Оператор == для порівняння двох рядків на рівність.

3. Оператор != для порівняння двох рядків на нерівність.
4. Оператор [] для доступу до символу за індексом.
5. Оператор << для виводу рядка на екран.

Додайте в клас також необхідні конструктори, деструктор та інші методи, які можуть знадобитись для роботи з рядками.

Напишіть програму, де ви використовуєте цей клас та перевірте роботу всіх перевантажених операторів. Створіть декілька об'єктів класу String і виконайте з ними операції конкатенації, порівняння на рівність, доступу до символу за індексом та виведення на екран.

Лістинг string.h:

```
#ifndef STRING_H
#define STRING_H

#include <iostream>

using namespace std;

class String
{
public:
    String();
    String(const char*);
    String(const String&);
    ~String();

    int getLength();
    void setLength(int);

    int countLength(const char*);
    void copyData(const char*, char*) const;

    String operator+(const String&) const;
    bool operator==(const String&) const;
    bool operator!=(const String&) const;

    char& operator[](int);
    const char& operator[](int) const;

    friend ostream& operator<<(ostream&, String&);
    friend istream& operator>>(istream&, String&);

private:
    char* data;
    int length;
};

#endif // STRING_H
```

Лістинг string.cpp:

```
#include "mystring/string.h"

String::String() {
    length = 0;
    data = new char[1];
    data[0] = '\0';
}

String::String(const char* str) : String() {
    if (str) {
        delete[] data;
        length = countLength(str);
        data = new char[length + 1];
        copyData(str, data);
    }
}

String::String(const String& other) {
    length = other.length;
    data = new char[length + 1];
    copyData(other.data, data);
}

String::~String() { delete[] data; }

int String::getLength() { return length; }
void String::setLength(int len) { length = len; }

int String::countLength(const char* str) {
    int len = 0;
    while (str[len] != '\0') ++len;
    return len;
}

void String::copyData(const char* orig, char* copy) const {
    int index = 0;
    while (orig[index] != '\0') {
        copy[index] = orig[index];
        ++index;
    }
    copy[index] = '\0';
}

String String::operator+(const String& other) const {
    String result;
    result.length = this->length + other.length;
    delete[] result.data;
    result.data = new char[result.length + 1];

    copyData(this->data, result.data);
    copyData(other.data, result.data + this->length);

    return result;
}

bool String::operator==(const String& other) const {
    if (length != other.length) return false;
```

```
    for (int i = 0; i < length; ++i) if (data[i] != other.data[i]) return
false;
    return true;
}

bool String::operator!=(const String& other) const { return !(*this ==
other); }

char& String::operator[](int index) {
    if (index < 0 || index >= length) return data[length];
    return data[index];
}

const char& String::operator[](int index) const {
    if (index < 0 || index >= length) return data[length];
    return data[index];
}

ostream& operator<<(ostream& out, String& str) {
    out << str.data;
    return out;
}

istream& operator>>(istream& in, String& str) {
    char buffer[1000];
    in >> buffer;

    delete[] str.data;
    str.setLength(0);
    str.setLength(str.countLength(buffer));

    str.data = new char[str.length + 1];
    str.copyData(buffer, str.data);

    return in;
}
```

Лістинг main.cpp:

```
#include "mystring/string.h"

int main()
{
    String s1, s2;

    cout << "Enter first string: ";
    cin >> s1;

    cout << "Enter second string: ";
    cin >> s2;

    cout << "You entered:" << endl;
    cout << "s1: " << s1 << endl;
    cout << "s2: " << s2 << endl;

    String s3 = s1 + s2;
    cout << "s1 + s2 = " << s3 << endl;
```

```
    cout << "s1 == s2 is " << (s1 == s2 ? "true" : "false") << endl;
    cout << "s1 != s2 is " << (s1 != s2 ? "true" : "false") << endl;

    if (s1.getLength() > 0) {
        cout << "First character of s1: " << s1[0] << endl;

        int index;
        char newChar;
        cout << "Enter index of character to change (0 to " << s1.getLength()
- 1 << "): ";
        cin >> index;
        cout << "Enter new character: ";
        cin >> newChar;

        if (index >= 0 && index < s1.getLength()) {
            s1[index] = newChar;
            cout << "s1 after modification: " << s1 << endl;
        } else {
            cout << "Invalid index!" << endl;
        }
    }
    return 0;
}
```

Результат:

```
Enter first string: Hello
Enter second string: World
You entered:
s1: Hello
s2: World
s1 + s2 = HelloWorld
s1 == s2 is false
s1 != s2 is true
First character of s1: H
Enter index of character to change (0 to 4): 2
Enter new character: 7
s1 after modification: He7lo
```