**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

**Інститут комп'ютерних наук та інформаційних технологій**
**Кафедра програмного забезпечення**


# ЗВІТ

До лабораторної роботи № 10
**На тему:** «Шаблони класів»
**З дисципліни:** «Об'єктно-орієнтоване програмування»

**Лектор:**
доц. кафедри ПЗ
Коротєєва Т. О.

**Виконав:**
ст. гр. ПЗ-16
Чаус О. М.

**Прийняв:**
асист. кафедри ПЗ
Дивак І. В.

« ____ » _____ 2022 р.

∑= _____ .....:

Львів-2022

**Тема роботи**: Шаблони класів
**Мета роботи**: Навчитись створювати шаблони класу та екземпляри шаблонів.

## Теоретичні відомості

Для оголошення шаблону функції використовується ключове слово template , далі в трикутних дужках записується тип параметру шаблону <typename T> або <class T> . У мові C++ прийнято називати типи параметрів шаблонів великою літерою T, але можна використовувати будь-який ідентифікатор.

```
template <typename T>

T& max(T& a, T& b)

{

    return (a > b) ? a : b;

}
```

Якщо потрібно кілька типів параметрів шаблону, то вони розділяються комами: template <typename T1, typename T2>
Коли компілятор зустрічає виклик шаблону функції, він копіює шаблон функції і замінює типи параметрів шаблону функції фактичними (переданими) типами даних. Функція з фактичними типами даних називається екземпляром шаблону функції (або «об'єктом шаблону функції»).

## Завдання для лабораторної роботи

Створити шаблон класу та продемонструвати його роботу за індивідуальним варіантом. Оформити звіт до лабораторної роботи. Звіт має містити варіант завдання, код розробленої програми, результати роботи програми (скріншоти), висновок.
Створити шаблон класу Set. Шаблон класу повинен давати можливість вивести всі елементи множини на екран, додавати, віднімати, здійснювати перетин множин (множина не може містити однакових елементів). Продемонструвати функціонал шаблону на створеному користувацькому типі Fraction – звичайний дріб. При порівнянні дробів дозволяється вважати 1/3 і 2/6 різними дробами.

## Хід роботи

Файл **set.h**:

```h
#ifndef SET_H
#define SET_H
#include <cstddef>

template<typename T>
class Set
{
private:
    T* m_array;
    std::size_t m_size;
public:
    Set();
    Set(const Set& secondSet);
    ~Set();
    bool addElement(T element);
    T * const begin() const;
    T * const end() const;
    bool contains(const T &element) const;
    Set& operator=(const Set& secondSet);
```

```
    Set operator+(const Set& secondSet) const;
    Set operator-(const Set& secondSet) const;
    Set intersect(const Set& secondSet) const;
    std::size_t getSize() const;
};

template<typename T>
Set<T>::Set() : m_array(nullptr), m_size(0) {};
template<typename T>
Set<T>::Set(const Set& secondSet)
{
    if(secondSet.m_size)
    {
        m_size = secondSet.m_size;
        m_array = new T[secondSet.m_size];
        for (std::size_t i = 0; i < m_size; i++)
        {
            m_array[i] = secondSet.m_array[i];
        }
    }
}
template<typename T>
Set<T>::~Set()
{
    delete [] m_array;
}
template<typename T>
bool Set<T>::addElement(T element)
{
    for(std::size_t i = 0; i < m_size; i++)
        if(element == m_array[i])
            return false;
    m_size++;
    T* array = new T[m_size];
    for (std::size_t i = 0; i < m_size - 1; i++)
        array[i] = m_array[i];
    array[m_size - 1] = element;
    delete[] m_array;
    m_array = array;
    return true;
}
template<typename T>

T * const Set<T>::begin() const
{
    return m_array;
}
template<typename T>
T * const Set<T>::end() const
{
    return m_array + m_size;
}
template<typename T>
bool Set<T>::contains(const T &element) const
{
    for(std::size_t i = 0; i < getSize(); i++)
        if(element == m_array[i])
            return true;
    return false;
}
template<typename T>
Set<T>& Set<T>::operator=(const Set<T>& secondSet)
{
    if (this != &secondSet)
    {
        delete[] m_array;
```

```
        m_size = secondSet.m_size;
        m_array = new T[secondSet.m_size];
        for (std::size_t i = 0; i < m_size; i++)
        {
            m_array[i] = secondSet.m_array[i];
        }
    }
    return *this;
}
template<typename T>
Set<T> Set<T>::operator+(const Set<T>& secondSet) const
{
    Set<T> newSet = *this;
    for (std::size_t i = 0; i < secondSet.m_size; i++)
    {
        bool isSame = false;
        for (std::size_t j = 0; j < m_size; j++)
        {
            if (m_array[j] == secondSet.m_array[i])
            {
                isSame = true;
                break;
            }
        }
        if (!isSame)
        {
            newSet.addElement(secondSet.m_array[i]);
        }
    }
    return newSet;
}
template<typename T>
Set<T> Set<T>::operator-(const Set<T>& secondSet) const
{
    Set<T> newSet;
    for (std::size_t i = 0; i < m_size; i++)
    {
        bool isSame = false;
        for (std::size_t j = 0; j < secondSet.m_size; j++)
        {
            if (m_array[i] == secondSet.m_array[j])
            {
                isSame = true;
                break;
            }
        }
        if (!isSame)
        {
            newSet.addElement(m_array[i]);
        }
    }
    return newSet;
}
template<typename T>
Set<T> Set<T>::intersect(const Set<T>& secondSet) const
{
    Set<T> newSet;
    std::size_t newSize = m_size;
    for (std::size_t i = 0; i < secondSet.m_size; i++)
    {
        bool isSame = false;
        for (std::size_t j = 0; j < m_size; j++)
        {
            if (m_array[j] == secondSet.m_array[i])
            {
                isSame = true;
```

```
                break;
            }
        }
        if (isSame)
        {
            newSet.addElement(secondSet.m_array[i]);
        }
    }
    return newSet;
}

template<typename T>
std::size_t Set<T>::getSize() const
{
    return m_size;
}


#endif // SET_H
```

Файл **fraction.h**:

```
#ifndef FRACTION_H
#define FRACTION_H
#include <cstddef>
#include <QTableWidgetItem>

class Fraction
{
private:
    int m_numerator;
    int m_denominator;
public:
    Fraction();
    Fraction(int num, int den);
    void print(QTableWidgetItem *item) const;
    bool operator==(Fraction secondFraction) const;
    int getNumerator() const;
    int getDenominator() const;
};


#endif // FRACTION_H
```

Файл **fraction.cpp**:

```
#include "fraction.h"


Fraction::Fraction() : m_numerator(1), m_denominator(1) {};
Fraction::Fraction(int num, int den) : m_numerator(num), m_denominator(den) {};
void Fraction::print(QTableWidgetItem *item) const
{
    item->setText(QString::number(m_numerator) + '/' + QString::number(m_denominator));
}
bool Fraction::operator==(Fraction secondFraction) const
{
    if(static_cast<double>(m_numerator) / m_denominator
                == static_cast<double>(secondFraction.m_numerator) /
secondFraction.m_denominator)
    {
        return true;
    }
    return false;
}

int Fraction::getNumerator() const
{
```

```
    return m_numerator;
}
int Fraction::getDenominator() const
{
    return m_denominator;
}
```

Файл **mainwindow.cpp**:

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include "set.h"
#include "fraction.h"

Set<Fraction> set1;
Set<Fraction> set2;
Set<Fraction> setRes;

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    ui->SetResultTable->resize(0, ui->SetResultTable->height());
    UpdateData();
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::UpdateData()
{
    ui->Set1Table->setColumnCount(set1.getSize());
    ui->Set1Table->resize(ui->Set1Table->columnCount() * 70, ui->Set1Table->height());
    int i = 0;
    for(Fraction &value : set1)
    {
        QTableWidgetItem *item = new QTableWidgetItem;
        item->setFlags(item->flags() ^ Qt::ItemIsEditable);
        value.print(item);
        ui->Set1Table->setItem(0, i, item);
        i++;
    }
    ui->Set2Table->setColumnCount(set2.getSize());
    ui->Set2Table->resize(ui->Set2Table->columnCount() * 70, ui->Set2Table->height());
    i = 0;
    for(Fraction &value : set2)
    {
        QTableWidgetItem *item = new QTableWidgetItem;
        item->setFlags(item->flags() ^ Qt::ItemIsEditable);
        value.print(item);
        ui->Set2Table->setItem(0, i, item);
        i++;
    }
}

void MainWindow::on_AddButton_clicked()
{
    if(ui->NumeratorLine->text().toInt() >= 0 && ui->DenominatorLine->text().toInt()
            && (ui->radioButton1->isChecked() || ui->radioButton2->isChecked()))
    {
        Fraction newElement(ui->NumeratorLine->text().toInt(), ui->DenominatorLine-
>text().toInt());
        Set<Fraction> *p = nullptr;
        if(ui->radioButton1->isChecked())
        {
```

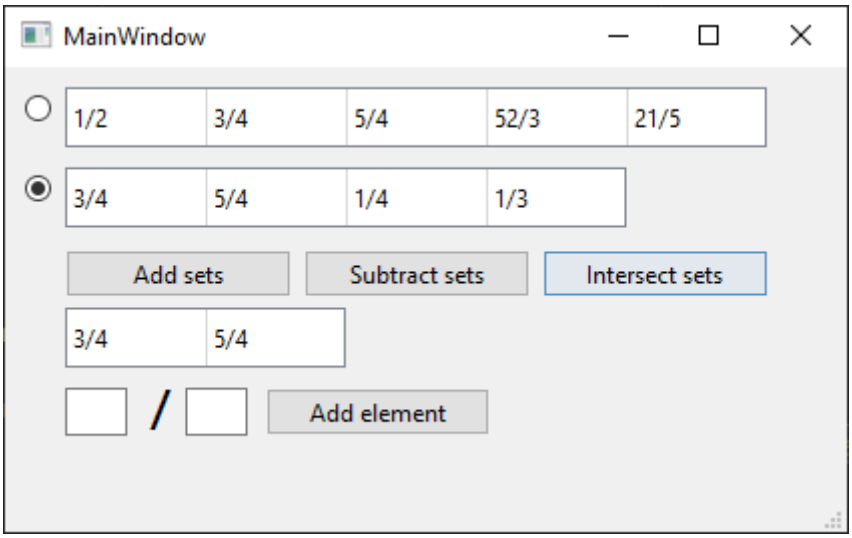
```
            p = &set1;
        }
        else if(ui->radioButton2->isChecked())
        {
            p = &set2;
        }
        if(p)
            if(p->addElement(newElement))
            {
                ui->NumeratorLine->clear();
                ui->DenominatorLine->clear();
            }
        UpdateData();
    }
}


void MainWindow::on_AddSetsButton_clicked()
{
    setRes = set1 + set2;
    ui->SetResultTable->setColumnCount(setRes.getSize());
    ui->SetResultTable->resize(ui->SetResultTable->columnCount() * 70, ui-
>SetResultTable->height());
    int i = 0;
    for(Fraction &value : setRes)
    {
        QTableWidgetItem *item = new QTableWidgetItem;
        item->setFlags(item->flags() ^ Qt::ItemIsEditable);
        value.print(item);
        ui->SetResultTable->setItem(0, i, item);
        i++;
    }
}

void MainWindow::on_SubtractButton_clicked()
{
    setRes = set1 - set2;
    ui->SetResultTable->setColumnCount(setRes.getSize());
    ui->SetResultTable->resize(ui->SetResultTable->columnCount() * 70, ui-
>SetResultTable->height());
    int i = 0;
    for(Fraction &value : setRes)
    {
        QTableWidgetItem *item = new QTableWidgetItem;
        item->setFlags(item->flags() ^ Qt::ItemIsEditable);
        value.print(item);
        ui->SetResultTable->setItem(0, i, item);
        i++;
    }
}


void MainWindow::on_IntersectButton_clicked()
{
    setRes = set1.intersect(set2);
    ui->SetResultTable->setColumnCount(setRes.getSize());
    ui->SetResultTable->resize(ui->SetResultTable->columnCount() * 70, ui-
>SetResultTable->height());
    int i = 0;
    for(Fraction &value : setRes)
    {
        QTableWidgetItem *item = new QTableWidgetItem;
        item->setFlags(item->flags() ^ Qt::ItemIsEditable);
        value.print(item);
        ui->SetResultTable->setItem(0, i, item);
        i++;
```

```
    }
}
```




*Зображення програми*

**Висновок:** : навчився створювати шаблони класу та екземпляри шаблонів.