Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Інститут комп’ютерних наук та інформаційних технологій

Кафедра ПЗ

Звіт

до лабораторної роботи №6

на тему «Перевантаження функцій і операцій, дружні функції, статичні члени класу»

з дисципліни “Об’єктно-орієнтоване програмування”

Виконав:

студент групи ПЗ-11

Солтисюк Дмитро

Перевірив:

доц. Коротєєва Т.О.

Львів

2022

Тема. Перевантаження функцій і операцій, дружні функції, статичні члени класу.
Мета. Навчитися використовувати механізм перевантаження функцій та операцій. Навчитися створювати та використовувати дружні функції. Ознайомитися зі статичними полями і методами та навчитися їх використовувати.

Завдання для лабораторної роботи:

На основі класу з попередньої лабораторної:

Перевантажити як мінімум три функції-члени з попереднього завдання.

Перевантажити операції згідно з варіантом. Для операцій, для яких не вказані символи, вибрати символи самостійно.

Створити дружні функції згідно з варіантом.

Створити статичні поля та статичні методи згідно з варіантом.

Продемонструвати розроблені можливості класу завдяки створеному віконному застосуванню.

Оформити звіт до лабораторної роботи.

1. Клас Drib – звичайний дріб.

Перевантажити операції, як функції члени:

Додавання

Віднімання

Множення

Ділення

Обертання дробу (операція "!").

Перевантажити операції, як дружні-функції:

Введення дробу з форми ("<<")

Виведення дробу на форму (">>")

Більше (">")

Менше ("<")

Рівне ("==").

Створити статичне поле, в якому б містилась інформація про кількість створених об’єктів, а також статичні функції для роботи з цим полем.

Теоретичні відомості:

C++ підтримує спеціальні засоби, які дозволяють перевизначити вже існуючі операції. Наприклад, для операції + можна ввести своє власне визначення, яке реалізує операцію додавання для об’єктів певного класу. Фактично перевизначення для операцій існує і в мові С. Так, операція + може використовувати як об’єкти типу int, так і об’єкти типу float. C++ розширює цю ідею.

Для визначення операції використовується функція, що вводиться користувачем. Тип функції визначається іменем класу, далі записується ключове слово **operator**, за яким слідує символ операції, в круглих дужках дається перелік параметрів, серед яких хоча б один типу клас.

Функції-операції мають бути нестатичними функціями-членами класу або мати мінімум один аргумент типу класу. За виключенням операції присвоєння всі перевизначені оператори наслідуються.

Дружньою функцією класу називається функція, яка сама не є членом класу, але має повні права на доступ до закритих та захищених елементів класу. Оскільки така функція не є членом класу, то вона не може бути вибрана з допомогою операторів (.) та (->), Дружня функція класу викликається звичайним способом. Для опису дружньої функції використовується ключове слово **friend**.

До тепер рахувалось, що дані в кожному об’єкті є власністю саме цього об’єкту та не використовуються іншими об’єктами цього класу. Та іноді приходиться слідкувати за накопиченням даних. Наприклад, необхідно з’ясувати скільки об’єктів даного класу було створено на даний момент та скільки з них існує. Статичні змінні-члени досяжні для всіх екземплярів класу. Це є компроміс між глобальними даними, які досяжні всім елементам програми, та даними, що досяжні тільки об’єктам даного класу. Статична змінні створюється в одному екземплярі для всіх об’єктів даного класу. Статичні змінні необхідно обов’язково ініціалізувати.

Статичні функції класу подібні до статичних змінних: вони не належать одному об’єкту, а знаходяться в області дії всього класу. Статичні функції-члени не мають вказівника this. Відповідно їх не можна оголосити як const. Статичні функції-члени не можуть звертатись до нестатичних змінних. До статичних функцій-членів можна звертатись з об’єкту їх класу, або вказавши повне ім’я, включаючи ім’я об’єкту.

Результат:

trianglesides.h

```
#pragma once

#include "triangleangles.h"
#include "triangleheights.h"
#include "trianglesides.h"
```

triangleangles.h

```
#pragma once

class TriangleAngles {
public:
  double abc;
  double bca;
  double cab;
};
```

triangleheights.h

```
#pragma once

class TriangleHeights {
public:
  double ak;
  double bk;
  double ck;
};
```

triangle.h

```
#pragma once

#include <QLineEdit>
#include <map>

#include "triangleangles.h"
#include "triangleheights.h"
#include "trianglesides.h"

#define TRIANGLE_SIDES_COUNT 3

class Triangle {
private:
  TriangleSides sides;
  static inline int trianglesCount = 0;
  double *sidesArr[TRIANGLE_SIDES_COUNT] = {&this->sides.a, &this->sides.b,
                                            &this->sides.c};

  bool isValid(TriangleSides sides);

public:
  Triangle(TriangleSides sides);
  Triangle();
```

```
  static int getTrianglesCount() { return Triangle::trianglesCount; };
  static void incrementTrianglesCount() { Triangle::trianglesCount++; };

  auto getSides() { return this->sides; };
  void setSides(TriangleSides sides) { this->sides = sides; };

  Triangle *operator+(const double increaseSidesBy);
  Triangle *operator*(const double increaseSidesTimes);

  double operator[](const int i);
  operator double();

  friend void operator<<(QLineEdit *out[0 + 2 + 1], Triangle &triangle);
  friend void operator>>(QLineEdit *in[0 + 2 + 1], Triangle &triangle);

  friend bool operator>(Triangle &tr1, Triangle &tr2);
  friend bool operator<(Triangle &tr1, Triangle &tr2);
  friend bool operator==(Triangle &tr1, Triangle &tr2);

  bool isRectangular();
  double area();
  double perimeter();
  TriangleAngles angles();
  TriangleHeights heights();
  TriangleSides increaseSidesBy(const double by);
  TriangleSides increaseSidesTimes(const double by);
};
```

triangle.cpp

```
#include "widget.h"
#include "triangle.h"
#include "triangleangles.h"
#include "trianglesides.h"

#include <QGridLayout>
#include <QMessageBox>

void Widget::onInputConfirm() {
  QLineEdit *sideInputs[] = {this->side_a, this->side_b, this->side_c};

  if (!this->triangle) {
    this->triangle = new Triangle();
  }

  sideInputs >> *this->triangle;

  emit valueChanged(this->triangle);
}

void Widget::onInputIncreaseBy() {
  *this->triangle + this->increaseSidesBy->text().toDouble();
  this->increaseSidesBy->clear();

  emit valueChanged(this->triangle);
}

void Widget::onInputIncreaseTimes() {
```

```
  (*this->triangle) * (this->increaseSidesTimes->text().toDouble());
  this->increaseSidesTimes->clear();

  emit valueChanged(this->triangle);
}

void Widget::onValueChange(Triangle *triangle) {
  TriangleSides sides = triangle->getSides();
  TriangleHeights heights = triangle->heights();
  TriangleAngles angles = triangle->angles();

  QLineEdit *sideInputs[] = {this->side_a, this->side_b, this->side_c};

  sideInputs << *this->triangle;

  this->triangleInstancesCount->setText(
      QString::number(triangle->getTrianglesCount()));
  this->area->setText(QString::number(triangle->area()));
  this->perimeter->setText(QString::number(triangle->perimeter()));
  this->isRectangular->setText(QVariant(triangle->isRectangular()).toString());

  this->heights->setText(QString("AK: %1\nBK: %2\nCK: %3")
                         .arg(heights.ak)
                         .arg(heights.bk)
                         .arg(heights.ck));

  this->angles->setText(QString("ABC: %1\nBCA: %2\nCAB: %3")
                        .arg(angles.abc)
                        .arg(angles.bca)
                        .arg(angles.cab));
}

Widget::Widget(QWidget *parent) : QWidget(parent) {
  QGridLayout *mainLayout = new QGridLayout;

  this->triangle = nullptr;

  this->side_a = new QLineEdit;
  this->side_a->setPlaceholderText("Side A length");

  this->side_b = new QLineEdit;
  this->side_b->setPlaceholderText("Side B length");

  this->side_c = new QLineEdit;
  this->side_c->setPlaceholderText("Side C length");

  this->increaseSidesBy = new QLineEdit;
  this->increaseSidesTimes = new QLineEdit;

  this->isRectangular = new QLineEdit;
  this->isRectangular->setReadOnly(true);

  this->area = new QLineEdit;
  this->area->setReadOnly(true);

  this->perimeter = new QLineEdit;
  this->perimeter->setReadOnly(true);

  this->angles = new QTextEdit;
```

```
    this->angles->setReadOnly(true);

    this->heights = new QTextEdit;
    this->heights->setReadOnly(true);

    this->confirmInput = new QPushButton("Enter");
    this->confirmIncreaseBy = new QPushButton("Increase by");
    this->confirmIncreaseTimes = new QPushButton("Increase times");

    this->triangleInstancesCount =
        new QLabel(QString::number(this->triangle->getTrianglesCount()));

    mainLayout->addWidget(this->side_a, 0, 0);
    mainLayout->addWidget(this->side_b, 0, 1);
    mainLayout->addWidget(this->side_c, 0, 2);
    mainLayout->addWidget(this->confirmInput, 1, 0, 1, 3);

    mainLayout->addWidget(new QLabel("Increase sides by:"), 2, 0);
    mainLayout->addWidget(this->increaseSidesBy, 2, 1);
    mainLayout->addWidget(this->confirmIncreaseBy, 2, 2);

    mainLayout->addWidget(new QLabel("Increase sides times:"), 3, 0);
    mainLayout->addWidget(this->increaseSidesTimes, 3, 1);
    mainLayout->addWidget(this->confirmIncreaseTimes, 3, 2);

    mainLayout->addWidget(new QLabel("Is rectangular:"), 4, 0);
    mainLayout->addWidget(this->isRectangular, 4, 1, 1, 2);

    mainLayout->addWidget(new QLabel("Area:"), 5, 0);
    mainLayout->addWidget(this->area, 5, 1, 1, 2);

    mainLayout->addWidget(new QLabel("Perimeter:"), 6, 0);
    mainLayout->addWidget(this->perimeter, 6, 1, 1, 2);

    mainLayout->addWidget(new QLabel("Angles:"), 7, 0);
    mainLayout->addWidget(this->angles, 7, 1, 1, 2);

    mainLayout->addWidget(new QLabel("Heights:"), 8, 0);
    mainLayout->addWidget(this->heights, 8, 1, 1, 2);

    mainLayout->addWidget(this->triangleInstancesCount, 9, 0);

    connect(this->confirmInput, &QPushButton::released, this,
            &Widget::onInputConfirm);

    connect(this->confirmIncreaseBy, &QPushButton::released, this,
            &Widget::onInputIncreaseBy);

    connect(this->confirmIncreaseTimes, &QPushButton::released, this,
            &Widget::onInputIncreaseTimes);

    connect(this, &Widget::valueChanged, &Widget::onValueChange);

    setLayout(mainLayout);
}
```

widget.h

```
#pragma once
```

```
#include "triangle.h"

#include <QLabel>
#include <QLineEdit>
#include <QPushButton>
#include <QTextEdit>
#include <QWidget>

class Widget : public QWidget {
  Q_OBJECT

public:
  Widget(QWidget *parent = nullptr);

private slots:
  void onInputConfirm();
  void onInputIncreaseBy();
  void onInputIncreaseTimes();
  void onValueChange(Triangle *triangle);

signals:
  void valueChanged(Triangle *triangle);

private:
  QLineEdit *side_a;
  QLineEdit *side_b;
  QLineEdit *side_c;
  QPushButton *confirmInput;

  QLineEdit *increaseSidesBy;
  QLineEdit *increaseSidesTimes;

  QPushButton *confirmIncreaseBy;
  QPushButton *confirmIncreaseTimes;

  QLabel *triangleInstancesCount;
  QLineEdit *area;
  QLineEdit *perimeter;
  QLineEdit *isRectangular;
  QTextEdit *angles;
  QTextEdit *heights;

  Triangle *triangle;
};
```

widget.cpp

```
#include "widget.h"
#include "triangle.h"
#include "triangleangles.h"
#include "trianglesides.h"

#include <QGridLayout>
#include <QMessageBox>

void Widget::onInputConfirm() {
  QLineEdit *sideInputs[] = {this->side_a, this->side_b, this->side_c};
```

```
  if (!this->triangle) {
    this->triangle = new Triangle();
  }

  sideInputs >> *this->triangle;

  emit valueChanged(this->triangle);
}

void Widget::onInputIncreaseBy() {
  *this->triangle + this->increaseSidesBy->text().toDouble();
  this->increaseSidesBy->clear();

  emit valueChanged(this->triangle);
}

void Widget::onInputIncreaseTimes() {
  (*this->triangle) * (this->increaseSidesTimes->text().toDouble());
  this->increaseSidesTimes->clear();

  emit valueChanged(this->triangle);
}

void Widget::onValueChange(Triangle *triangle) {
  TriangleSides sides = triangle->getSides();
  TriangleHeights heights = triangle->heights();
  TriangleAngles angles = triangle->angles();

  QLineEdit *sideInputs[] = {this->side_a, this->side_b, this->side_c};

  sideInputs << *this->triangle;

  this->triangleInstancesCount->setText(
      QString::number(triangle->getTrianglesCount()));
  this->area->setText(QString::number(triangle->area()));
  this->perimeter->setText(QString::number(triangle->perimeter()));
  this->isRectangular->setText(QVariant(triangle->isRectangular()).toString());

  this->heights->setText(QString("AK: %1\nBK: %2\nCK: %3")
                         .arg(heights.ak)
                         .arg(heights.bk)
                         .arg(heights.ck));

  this->angles->setText(QString("ABC: %1\nBCA: %2\nCAB: %3")
                        .arg(angles.abc)
                        .arg(angles.bca)
                        .arg(angles.cab));
}

Widget::Widget(QWidget *parent) : QWidget(parent) {
  QGridLayout *mainLayout = new QGridLayout;

  this->triangle = nullptr;

  this->side_a = new QLineEdit;
  this->side_a->setPlaceholderText("Side A length");

  this->side_b = new QLineEdit;
```

```
this->side_b->setPlaceholderText("Side B length");

this->side_c = new QLineEdit;
this->side_c->setPlaceholderText("Side C length");

this->increaseSidesBy = new QLineEdit;
this->increaseSidesTimes = new QLineEdit;

this->isRectangular = new QLineEdit;
this->isRectangular->setReadOnly(true);

this->area = new QLineEdit;
this->area->setReadOnly(true);

this->perimeter = new QLineEdit;
this->perimeter->setReadOnly(true);

this->angles = new QTextEdit;
this->angles->setReadOnly(true);

this->heights = new QTextEdit;
this->heights->setReadOnly(true);

this->confirmInput = new QPushButton("Enter");
this->confirmIncreaseBy = new QPushButton("Increase by");
this->confirmIncreaseTimes = new QPushButton("Increase times");

this->triangleInstancesCount =
    new QLabel(QString::number(this->triangle->getTrianglesCount()));

mainLayout->addWidget(this->side_a, 0, 0);
mainLayout->addWidget(this->side_b, 0, 1);
mainLayout->addWidget(this->side_c, 0, 2);
mainLayout->addWidget(this->confirmInput, 1, 0, 1, 3);

mainLayout->addWidget(new QLabel("Increase sides by:"), 2, 0);
mainLayout->addWidget(this->increaseSidesBy, 2, 1);
mainLayout->addWidget(this->confirmIncreaseBy, 2, 2);

mainLayout->addWidget(new QLabel("Increase sides times:"), 3, 0);
mainLayout->addWidget(this->increaseSidesTimes, 3, 1);
mainLayout->addWidget(this->confirmIncreaseTimes, 3, 2);

mainLayout->addWidget(new QLabel("Is rectangular:"), 4, 0);
mainLayout->addWidget(this->isRectangular, 4, 1, 1, 2);

mainLayout->addWidget(new QLabel("Area:"), 5, 0);
mainLayout->addWidget(this->area, 5, 1, 1, 2);

mainLayout->addWidget(new QLabel("Perimeter:"), 6, 0);
mainLayout->addWidget(this->perimeter, 6, 1, 1, 2);

mainLayout->addWidget(new QLabel("Angles:"), 7, 0);
mainLayout->addWidget(this->angles, 7, 1, 1, 2);

mainLayout->addWidget(new QLabel("Heights:"), 8, 0);
mainLayout->addWidget(this->heights, 8, 1, 1, 2);

mainLayout->addWidget(this->triangleInstancesCount, 9, 0);
```

```
    connect(this->confirmInput, &QPushButton::released, this,
            &Widget::onInputConfirm);

    connect(this->confirmIncreaseBy, &QPushButton::released, this,
            &Widget::onInputIncreaseBy);

    connect(this->confirmIncreaseTimes, &QPushButton::released, this,
            &Widget::onInputIncreaseTimes);

    connect(this, &Widget::valueChanged, &Widget::onValueChange);

    setLayout(mainLayout);
}
```

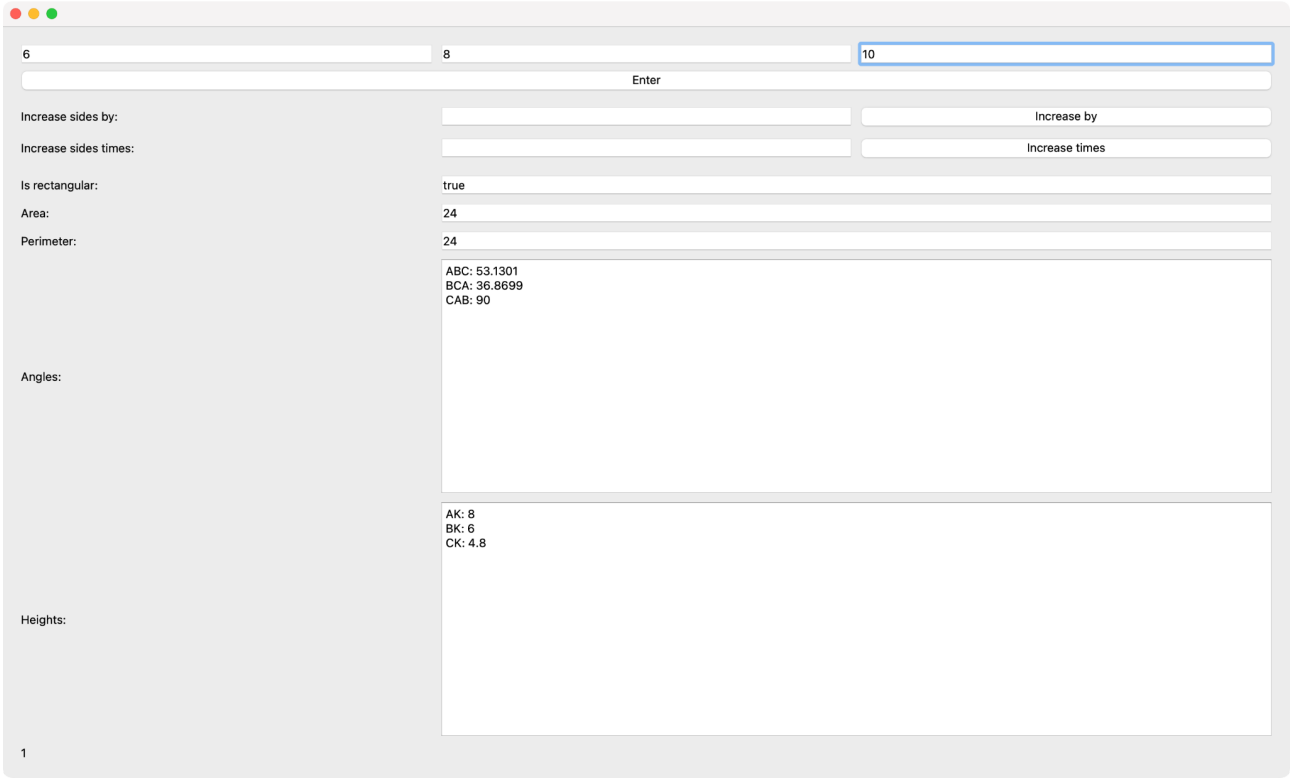

Рис.1. Робота програми

Висновок:

У ході лабораторної роботи №6 я навчився використовувати механізм перевантаження функцій та операцій, створювати та використовувати дружні функції, статичні поля і методи.