**Міністерство освіти і науки України**

**Запорізький національний технічний університет**

Кафедра програмних засобів

**ЗВІТ**

З лабораторної роботи №4

з дисципліни «ЯПЗтаТ»

на тему:

**«ОБЧИСЛЕННЯ ОБЁЄКТНО-ОРІЄНТОВАНИХ МЕТРИК»**

|  |  |
| --- | --- |
| Виконав: |  |
| студент КНТ-415 | Р.О. Шлома |
|  |  |
| Прийняв: |  |
| доцент | Т. І. Каплієнко |
|  |  |
| 2018 | |

Мета

Вивчити основні об’єктно-орієнтовані метрики програмних систем та реалізувати їх із застосуванням мови С++ та Qt.

Завдання до роботи

1. Ознайомитися з теоретичними відомостями щодо об’єктно-орієнтованих метрик програмних систем.
2. Реализувати на мові C++ програму розрахунку об’єктно-орієнтованих метрик.
3. Виконати аналіз отриманих результатів.
4. Оформити звіт та відповісти на контрольні питання.

Виконання завдань до роботи

*Код розробленої програми*

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

#include <QMessageBox>

#include <QTextStream>

*static* QString code = "";

*static* QStringList lis, Lclss, Rclss;

MainWindow::**MainWindow**(QWidget \*parent) :

QMainWindow(parent),

ui(*new* Ui::MainWindow)

{

ui->setupUi(*this*);

}

MainWindow::~***MainWindow***()

{

*delete* ui;

}

float MainWindow::**SrKolvoStrokMetodov**(QString code)

{

*//* *записуємо* *регулярне* *вираження* *та* *ділимо* *текст*

QRegExp \*rx = *new* QRegExp("(void|int|string|float|double|byte|bool)\\s+(\\w+::)?\\w+\\(.\*\\)");

rx->setMinimal(*true*);

QStringList funs = code.split(\*rx, QString::*SkipEmptyParts*);

double lines = 0;

QString fun;

*//* *розраховуємо* *загальну* *кількість* *рядків*

*for*(int i = 1; i < funs.count(); i++) {

lines += funs[i].count("\n");

}

*//* *розраховуємо* *середнє* *значення*

*return* funs.count() == 1 ? 0 : lines / (funs.count() - 1);

}

float MainWindow::**KolvoStrokMetodov**(QString code)

{

QStringList lst = code.split(QRegExp("(void|int|string|float|double|byte)[A-Za-z0-9]\*"), QString::*SkipEmptyParts*);

float s1 = lst.count()-1;

*return* s1;

}

float MainWindow::**SrKolvoStrokModuley**(QString code)

{

QStringList lst = code.split(QRegExp("//module"), QString::*SkipEmptyParts*);

int s = 0;

*for*(int i = lst.count() - 1; i >= 0; --i)

{

*const* QString& item = lst[i];

s = s + item.count("\n") - 1;

}

float s1 = (float) s / (lst.count());

*return* s1;

}

float MainWindow::**KolvoStrokModuley**(QString code)

{

QStringList lst = code.split(QRegExp("//module"), QString::*SkipEmptyParts*);

float s1 = lst.count() - 1;

*return* s1;

}

float MainWindow::**KolvoStrokModuleyInc**(QString code)

{

QStringList lst = code.split(QRegExp("#include"), QString::*SkipEmptyParts*);

float s1 = lst.count()-1;

*return* s1;

}

float MainWindow::**SrKolvoStrokClassov**(QString code)

{

QStringList lst = code.split(QRegExp("class "), QString::*SkipEmptyParts*);

int s = 0;

*for*(int i = lst.count()-1; i >= 0; --i)

{

*const* QString& item = lst[i];

s=s+item.count("\n")-1;

}

float s1 = (float) s/(lst.count());

*return* s1;

}

float MainWindow::**KolvoStrokClassov**(QString code)

{

QStringList lst = code.split(QRegExp("class "), QString::*SkipEmptyParts*);

float s1 = lst.count() - 1;

*return* s1;

}

int MainWindow::**NORM**(QString code)

{

QStringList lst = code.split(QRegExp("[A-Za-z0-9]\*[.]{1,1}[A-Za-z0-9]\*[(]\*[)]"), QString::*SkipEmptyParts*);

int s = -1;

*for*(int i = 0; i < lst.count(); i++) {

s++;

}

*return* s;

}

void MainWindow::**RFC**(QString code)

{

QStringList lst = code.split(QRegExp("[:.][A-Za-z0-9]\*[(]\*[A-Za-z0-9 ]\*[)]"), QString::*SkipEmptyParts*);

int s = -1;

*for*(int i = 0; i < lst.count(); i++) {

s++;

}

ui->label\_RFC->setText(QString::number(s));

}

void MainWindow::**MWC**(QString code)

{

ui->label\_WMC->setText("");

QStringList lst = code.split(QRegExp("class "), QString::*SkipEmptyParts*);

int s = 0;

*for*(int i = lst.count()-1; i >= 0; --i) {

*const* QString& item = lst[i];

QStringList list = item.split("){", QString::*SkipEmptyParts*);

s=0;

*for*(int j = list.count()-1; j >= 0; --j) {

*const* QString& ite = list[j];

*if* ((ite.count("{")>=ite.count("}"))&&(ite.count("}")!=0))

s=s+1;

}

ui->label\_WMC->setText( ui->label\_WMC->text()+" "+QString::number(s));

}

}

void MainWindow::**LCOM**(QString code)

{

int P=0, Q=0;

int m;

QString coual="";

QStringList lst = code.split(QRegExp("class "), QString::*SkipEmptyParts*);

int s = 0;

*for*(int i = lst.count()-1; i >= 0; --i) {

P=0;

Q=0;

*const* QString& item = lst[i];

QStringList list = item.split("){", QString::*SkipEmptyParts*);

s=0;

*for*(int j = list.count()-1; j >= 0; --j) {

*const* QString& ite = list[j];

QStringList lt = ite.split(QRegExp("(void|int|string|float|double|byte)[A-Za-z0-9]\*"), QString::*SkipEmptyParts*);

lt.removeDuplicates();

m = lt.count();

*//* *lt.removeAt(lt.count()-1);*

*for*(int g = lt.count()-1; g >= 1; --g) {

*const* QString& ir = lt[g];

QStringList l = ir.split(" ");

QString dd="+-\*=:/%()";

*for*(int c = l.count()-1; c >= 1; --c) {

*const* QString& r = l[c];

*if*(dd.count(r)==0)

coual=coual+r;

coual=coual+"++";

}

}

}

QStringList vars = coual.split("++");

*for*(int h = vars.count()-1; h >= 1; --h) {

*const* QString& qw = vars[h];

int y=0;

*for*(int k = h; k >= 0; --k) {

*const* QString& qw1 = vars[k];

*if* (qw==qw1)

{P=P+1; *break*;}

*else*

y=y+1;

*if* ((k==0)&&(y==vars.count()-1-h))

Q=Q+1;

}

}

float C = 0;

float f1=1;

float f2=1;

*for* (int z=1; z<=m; z++)

f1=f1\*z;

*for* (int z=1; z<=m-2; z++)

f2=f2\*z;

C = sqrt(f1/2/f2);

C = (P-Q)/C;

ui->label\_LCOM->setText( ui->label\_LCOM->text()+" "+QString::number(C));

}

}

void MainWindow::**NOC**(QString code)

{

ui->label\_NOC->setText("");

QStringList lst = code.split(QRegExp("class "), QString::*SkipEmptyParts*);

lst.removeFirst();

QStringList lis, Lclss, Rclss;

*for* (int g = 0; g<lst.count(); g++) {

*const* QString& item = lst[g].trimmed();

QString St ="";

*for*(int i=0; i<item.count(); i++)

*if* (item[i]!='\n')

St=St+QString(item[i]);

*else*

*break*;

lis<<St;

}

*for* (int g = 0; g<lis.count(); g++) {

*const* QString& item = lis[g].trimmed();

QString St ="";

*for*(int i=0; i<item.count(); i++)

*if* (item[i]!=' ')

St=St+QString(item[i]);

*else* *break*;

Rclss<<St;

}

*for* (int g = 0; g<lis.count(); g++) {

*const* QString& item = lis[g].trimmed();

QString St ="";

lst = item.split(QRegExp("public|protected|private "), QString::*SkipEmptyParts*);

*for*(int i=0; i<lst.count(); i++) {

St = St+lst[i];

}

Lclss<<St;

}

*for* (int g = 0; g<Rclss.count(); g++) {

Lclss[g].replace(0,Rclss[g].count(), "");

Lclss[g] = Lclss[g].trimmed();

}

*for* (int g = 0; g<Rclss.count(); g++) {

Rclss[g].remove(":");

int s = 0;

*for* (int i = 0; i < Rclss.count(); i++)

{s=s+Lclss[i].count(Rclss[g]);}

ui->label\_NOC->setText( ui->label\_NOC->text()+" "+Rclss[g]+" "+QString::number(s));

}

}

void MainWindow::**DIT**(QString code)

{

lis.clear();

Lclss.clear();

Rclss.clear();

QStringList lst = code.split(QRegExp("class "), QString::*SkipEmptyParts*);

lst.removeFirst();

*for* (int g = 0; g < lst.count(); g++) {

*const* QString& item = lst[g].trimmed();

QString St = "";

*for*(int i = 0; i < item.count(); i++)

*if* (item[i] != '\n')

St = St + QString(item[i]);

*else* *break*;

lis<<St;

}

*for* (int g = 0; g < lis.count(); g++) {

*const* QString& item = lis[g].trimmed();

QString St = "";

*for*(int i=0; i<item.count(); i++)

*if* (item[i]!=' ')

St = St + QString(item[i]);

*else* *break*;

Rclss<<St;

}

*for* (int g = 0; g < lis.count(); g++) {

*const* QString& item = lis[g].trimmed();

QString St ="";

lst = item.split(QRegExp("public|protected|private "), QString::*SkipEmptyParts*);

*for*(int i=0; i<lst.count(); i++)

St = St+lst[i];

Lclss<<St;

}

*for* (int g = 0; g < Rclss.count(); g++) {

Lclss[g].replace(0,Rclss[g].count(),"");

Lclss[g] = Lclss[g].trimmed();

}

*for* (int g = 0; g < Rclss.count(); g++) {

Rclss[g].remove(":");

Lclss[g].remove(",");

}

int\*\* mass = *new* int\*[Rclss.count()+1];

*for*(int i = 0; i <= Rclss.count(); i++)

mass[i]= *new* int [Rclss.count()+1];

*for* (int g = 0; g < Rclss.count(); g++)

*for* (int j = 0; j < Rclss.count(); j++) {

mass[g][j] = 0;

*if* (g != j) {

mass[g][j] = Lclss[g].count(Rclss[j]);

}

}

int pst = 0;

int prw = -1;

int sum =0;

int max = 0;

ui->label\_DIT->setText("");

*for* (int i = 0; i<Rclss.count(); i++) {

prw = i;

pst = 0;

max = 0;

sum = 0;

*while* (pst < Rclss.count()) {

int k;

*if* (mass[prw][pst]==1) {

*if* (prw == i)

k = pst;

mass[prw][pst] = 0;

prw = pst;

pst = 0;

sum = sum+1;

}

*else* pst=pst+1;

*if* (max<sum)

max=sum;

*if* ((pst==Rclss.count())&&(prw!=i)) {

prw = i;

pst=0;

sum=0;

}

*if* ((pst==Rclss.count())&&(prw==i))

*break*;

}

*for* (int g = 0; g<Rclss.count(); g++)

*for* (int j = 0; j<Rclss.count(); j++) {

mass[g][j]=0;

*if* (g!=j)

mass[g][j] = Lclss[g].count(Rclss[j]);

}

ui->label\_DIT->setText(ui->label\_DIT->text() + " " + Rclss[i] + " " + QString::number(max));

}

}

void MainWindow::**on\_openfile\_clicked**()

{

QString fileName = QFileDialog::getOpenFileName(*this*, tr("Open File"), "",tr("C++ Files (\*.cpp \*.h)"));

*if* (fileName != "") {

QFile file(fileName);

*if* (!file.*open*(QIODevice::*ReadOnly*)) {

QMessageBox::critical(*this*, tr("Error"), tr("Could not open file"));

*return*;

}

QTextStream in(&file);

ui->plainTextEdit->setText(in.readAll());

file.*close*();

}

code = ui->plainTextEdit->toPlainText();

}

void MainWindow::**on\_Metrics\_clicked**()

{

QString code = ui->plainTextEdit->toPlainText();

*if*(code == "")

*return*;

ui->label\_SrKSMet->setText(QString::number(SrKolvoStrokMetodov(code)));

ui->label\_SrKSMod->setText(QString::number(SrKolvoStrokModuley(code)));

ui->label\_SrKSCl->setText(QString::number(SrKolvoStrokClassov(code)));

MainWindow::MWC(code);

MainWindow::DIT(code);

MainWindow::NOC(code);

MainWindow::RFC(code);

MainWindow::LCOM(code);

ui->label\_NORM->setText(QString::number(NORM(code)));

ui->Kol\_Met->setText(QString::number(KolvoStrokMetodov(code)));

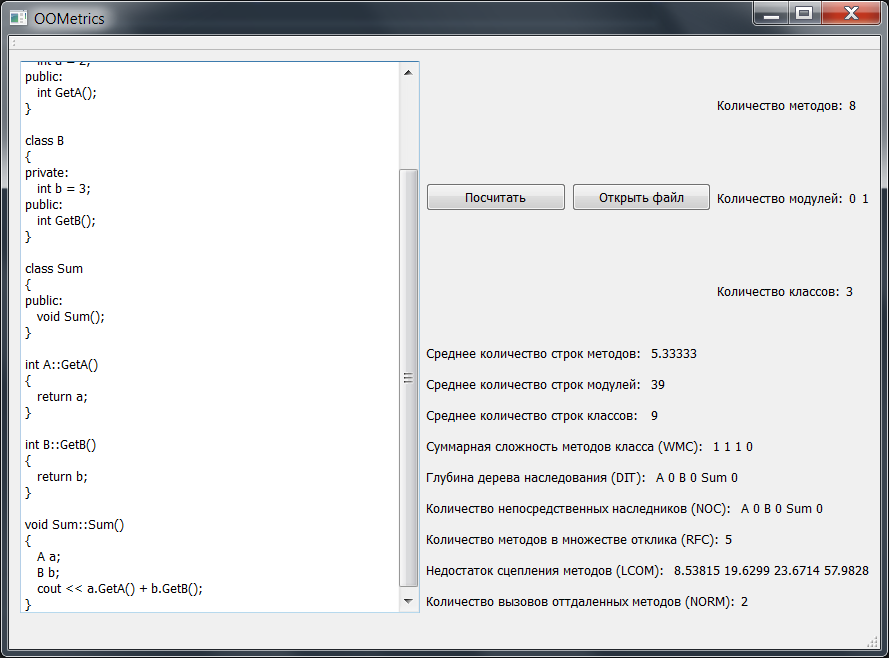
ui->Kol\_Mod->setText(QString::number(KolvoStrokModuley(code)));

ui->Kol\_Class->setText(QString::number(KolvoStrokClassov(code)));

ui->Include->setText(QString::number(KolvoStrokModuleyInc(code)));

}

*Результат роботи програми*



Висновок

Вивчив основні об’єктно-орієнтовані метрики програмних систем.