Міністерство освіти і науки України

Запорізький національний технічний університет

кафедра програмних засобів

ЗВІТ

З лабораторної роботи №4

З дисципліни «Якість програмного забезпечення та тестування» з теми:

**«ОБЧИСЛЕННЯ ОБ’ЄКТНО-ОРІЄНТОВАНИХ МЕТРИК»**

Виконав:

студент групи КНТ-115 Ю.О. Лукашенко

Прийняла:

доцент, к.т.н Н.О.Миронова

2018

1. Мета роботи

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

1. Завдання до роботи

Реалізувати на мові C++ програму розрахунку об’єктно-орієнтованих метрик: середня кількість рядків для функцій (методів); середня кількість рядків, що містять вихідний код для функцій (методів); середня кількість рядків для модулів; середня кількість рядків для класів; кількість викликів віддалених методів; відгук на клас; зважена насиченість класу; недолік зв’язності методів; метрики *WMC, DIT, NOC, RFC, LCOM*.

1. Загальні об’єктно-орієнтовані метрики.

До найбільш поширених *ОО* метрик відносяться:

Метрика *WMC (Weighted methods per class),* сумарна складність всіх методів класу.

Метрика *DIT (Depth of Inheritance tree*) – глибина дерева спадкування, чим більше, тим краще, так як при більшій глибині збільшується абстракція даних, зменшується насиченість класу методами, однак при досить великій глибині сильно зростає складність розуміння і написання програми.

Метрика *NOC (Number of children)* – кількість нащадків (безпосередніх), чим більше, тим вище абстракція даних.

Метрика *RFC (Response for a class)* – *RFC=/RS/,* де *RS* – відповідна множина класу, тобто безліч методів, які можуть бути потенційно викликані методом класу у відповідь на дані, отримані об'єктом класу. Чим більше *RFC*, тим складніше тестування та налагодження.

Метрика *LCOM* *(Lack of cohesion in Methods*) – недолік зчеплення методів. Якщо визначимо *P* – безліч пар методів, що не мають спільних змінних; *Q* – безліч пар методів, що мають загальні змінні,тоді .

1. Хід роботи

***4.4.1 Файл mainwindow.cpp***

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include <QFileDialog>

#include <QMessageBox>

#include <QTextStream>

MainWindow::MainWindow(QWidget \*parent) :

QMainWindow(parent),

ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

MainWindow::~*MainWindow*()

{

delete ui;

}

QString code;

double SrKolvoStrokMetodov(QString code)

{

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

int s = 0;

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

{

const QString& item = lst[i];

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

}

double s1= (double) s/(lst.count())-1;

return s1;

}

double KolvoStrokMetodov(QString code)

{

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

double s1 = lst.count()-1;

return s1;

}

double 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;

}

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

return s1;

}

double KolvoStrokModuley(QString code)

{

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

double s1=lst.count()-1;

return s1;

}

double KolvoStrokModuleyInc(QString code)

{

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

double s1=lst.count()-1;

return s1;

}

double 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;

}

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

return s1;

}

double KolvoStrokClassov(QString code)

{

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

double s1=lst.count()-1;

return s1;

}

int NORM(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];

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

s=s+item.count(") ")-item.count("this.")-item.count(" ) ");

}

return s;

}

int RFC(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];

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

s=s+item.count(") ")-item.count(" ) ");

}

return s;

}

void MainWindow::MWC(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];

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);

//QStringList::iterator t = list.begin();

// s=s+item.count(") ")-item.count(" ) ");

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;

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;

}

}

double C = 0;

double f1=1;

double 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(QString::number(Q));

}

}

void MainWindow::NOC(QString code)

{

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

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

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(QString::number(s));

}

}

void MainWindow::DIT(QString code)

{

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

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(":");

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(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();

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);

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

MainWindow::LCOM(code);

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

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

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

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

}

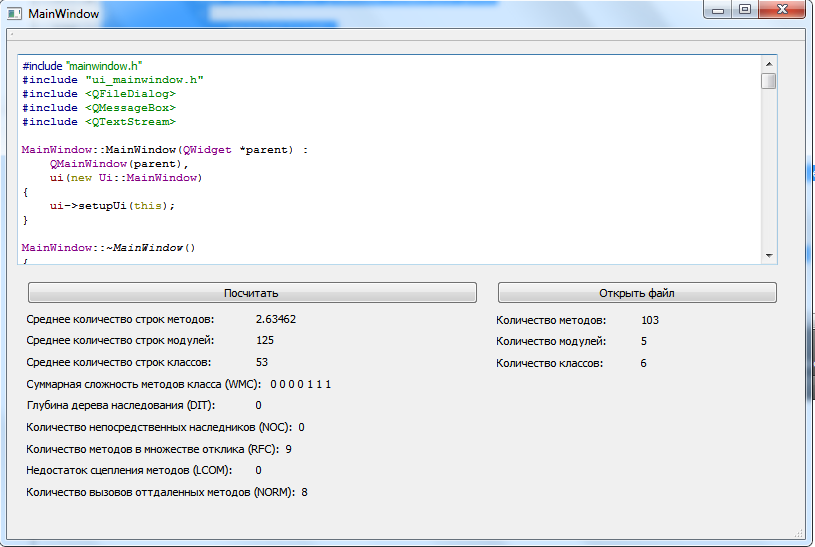


Рисунок 4.1 – Виконання програми

1. Висновки

В ході виконання лабораторної роботи, я засвоїв основні метрики коду програмних систем та реалізував їх із застосуванням мови С++ та Qt.